A Collection of the AIAA MODELING AND SIMULATION TECHNOLOGIES CONFERENCE Technical Papers

Denver, Colorado
14–17 August 2000



For permission to copy or republish, contact:
American Institute of Aeronautics and Astronautics
1801 Alexander Bell Drive
Suite 500
Reston, VA 20191–4344

A Collection of
Technical Papers

AIAA MODELING AND SIMULATION TECHNOLOGIES CONFERENCE

Denver, Colorado
14–17 August 2000

Copyright © 2000 by
American Institute of Aeronautics and Astronautics

ISBN
1-56347-378-X

All rights reserved. No part of this volume may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without permission from the publisher.

Printed in the U.S.A.

TABLE OF CONTENTS

14-17 August 2000

AIAA Modeling and Simulation Technologies Conference and Exhibiton

AIAA PAPER NUMBER	TITLE *AUTHOR(S)*	
2000-3974	**The Use of Full Flight Simulators & Flight Training Devices in the Fields of Helicopter Training, Testing, & Checking** *B. Hampson*	1
2000-4073	**G-Cueing for Fighter Simulation Training** *M. O. Sabourin and V. Sondermeyer*	9
2000-4075	**Development Issues of a Sustainable-G Dynamic Flight Simulator** *C. Spenny, B. Liebst, T. Chelette, C. Folescu, and J. Sigda*	18
2000-4076	**Simulated Landings in Turbulance - Motion, Predictive Modeling, and Psychometric Aspects** *E. Svensson and M. Angelborg-Thanderz*	30
2000-4077	**Standard Flight Simulator as a Spatial Disorientation Training Device** *C. Szczepanski, F. Skibniewski, O. Truszczynski, and M. Gasik*	41
2000-4080	***Low-Cost Simulation Solutions for Aerospace Projects*** *F. Cornelissen, A. Broeils, L. Bremer, and L. Gale*	50
2000-4084	**Draper Station Simulation - A Standard Architecture for Space Vehicle Simulation** *J. W. Jang, N. Bedrossian, J. Zoss, J. Templeton, and E. McCants*	60
2000-4085	**Simulators for the X-38/CRV Parafoil and Re-Entry Phases** *F. Delgado and M. Arcioni*	71

TABLE OF CONTENTS

14-17 August 2000

AIAA Modeling and Simulation Technologies Conference and Exhibiton

AIAA PAPER NUMBER	TITLE / *AUTHOR(S)*	
2000-4086	**Atmospheric Re-Entry Modeling and Simulation: Application to the Crew Return Vehicle** R. Costa, J. Silva, S. Wu, Q. Chu, and J. Mulder	78
2000-4087	**Advanced Space Robotics Simulation for Training and Operations** X. Cyril, G. Jaar, and J. St-Pierre	89
2000-4089	**Visualization, Interrogation, and Interpretation of Computed Flow Fields - Numerical Experiments** A. Hanneman and R. Henderson	95
2000-4090	**Simulation of Aircraft Landing Gears with a Nonlinear Dynamic Finite Element Code** K. Lyle, K. Jackson, and E. Fasanella	104
2000-4091	**State Representation and Simulation of 2D Turbulent Wind** N. Imbert, Ph. Mouyon, and G. Montseny	114
2000-4093	**Preparing Meteorlogical Data for Simulations using SEDRIS** L. Hembree, R. Cox, and V. Pastor	120
2000-4094	**Dynamic Hysteresis Modeling** P. Dahl and J. Ly	129
2000-4095	**Towards Realistic Forest Machine Simulators** E. Freund, M. Kraemer, and J. Rossmann	142

TABLE OF CONTENTS

14-17 August 2000

AIAA Modeling and Simulation Technologies Conference and Exhibiton

AIAA PAPER NUMBER	TITLE *AUTHOR(S)*	
2000-4169	**The Development of a Motion System for Cabin Emergency Evacuation Trainers** *P. Piatkiewitz*	153
2000-4170	**ATTAS Ground Based System Simulator - An Update** *S. Klaes*	160
2000-4173	**The Pilatus Engineering Flight Simulator** *L. Marchesini*	168
2000-4174	**A Description of the Cockpit Motion Facility and the Research Flight Deck Simulator** *M. Smith*	176
2000-4176	**DESDEMONA: Advanced Disorientation Trainer and (SUSTAINED-G) Flight Simulator** *W. Bles, J. Hosman, and B. de Graaf*	187
2000-4177	**Simulation of the Dynamics of Space Systems using Object-Oriented Modeling** *B. Min, A. Misra, X. Cyril, V. Modi, and C. de Silva*	191
2000-4178	**A New Method to Obtain a Unique Solution in System Identification** *M. Misawa*	202
2000-4179	**Dynamic Modeling and Stability Analysis of Z-SAT** *A. Alasty, K. Daneshjou, and S. Rezaei*	208

TABLE OF CONTENTS

14-17 August 2000

AIAA Modeling and Simulation Technologies Conference and Exhibiton

AIAA PAPER NUMBER	TITLE / AUTHOR(S)	
2000-4180	**Optimisation of a Reusable Launch Vehicle Concept** *M. Tetlow, G. Schneider, and U. Schoettle*	213
2000-4181	**The Life Cycle Application of Intelligent Software Modeling for the First Materials Science Research Rack** *A. Rice, F. Parris, and P. Nerren*	224
2000-4182	**Control Boom Design of Z-SAT** *K. Daneshjou, A. Alasty, and S. Rezaei*	233
2000-4184	**Set Up and Utilization of a Spacecraft Mechanics Software Tool to Simulate the Global Positioning System** *A. Al-Shehabi*	237
2000-4185	**Hybrid-System Simulation for National Airspace System Safety Analysis** *A. Pritchett, S. Lee, D. Huang, and D. Goldsman*	244
2000-4186	**Generic Development Environment For INS Algorithms** *J. Reiner*	255
2000-4188	**A Simulation Model of an Uninhabited Aircraft** *M. Perhinschi*	264
2000-4284	**Mathematical Models in Support of Hardware In The Loop Simulations** *R. Gorecki*	274

TABLE OF CONTENTS

14-17 August 2000

AIAA Modeling and Simulation Technologies Conference and Exhibiton

AIAA PAPER NUMBER	TITLE *AUTHOR(S)*	
2000-4285	**Using Simulation to Optimize Ski Jump Ramp Profiles for STOVL Aircraft** *G. Imhof and W. Schork*	282
2000-4286	**A Comparative Assessment Of Various Multipath Models For Use In Missile Simulation Studies** *D. Bucco and Y. Hu*	290
2000-4288	**Missle Aerodynamic Identification using Mixed EKF and EBM** *A. Mohammadi and M. Massoumnia*	300
2000-4289	**Required Simulation Attributes for Utility** *J. Heidt and B. O'Neal*	308
2000-4290	**Variable Windows and Forcing Function Algorithms for Hardware In The Loop Simulations** *R. Gorecki*	317
2000-4291	**Investigation of Mathematical Models of Otolith Organs for Human Centered Motion Cueing Algorithms** *R. Telban, F. Cardullo, and L. Guo*	326
2000-4292	**In-Flight Estimation of Pilots' Acceleration Sensitivity Thresholds** *V. Rodchenko, A. White, and S. Boris*	337
2000-4293	**Measuring the Realism of Motion in Flight Simulators** *H. van der Steen*	345

TABLE OF CONTENTS

14-17 August 2000

AIAA Modeling and Simulation Technologies Conference and Exhibiton

AIAA PAPER NUMBER	TITLE / AUTHOR(S)	
2000-4295	**Theoretical Approach to Estimation of Acceleration Effects on Piloting** *L. Zaichik, V. Rodchenko, Y. Yashin, I. Rufov, and A. White*	351
2000-4296	**The Effect of Simulator Motion on Pilot Training and Evaluation** *T. Go, J. Burki-Cohen, and N. Soja*	359
2000-4297	**Move or Not to Move? Continuous Question** *C. Szczepanski and D. Leland*	368
2000-4299	**Advanced Helicopter Flight and Aeroelastic Simulation Based on General Purpose Multibody Code** *T. Hablowetz*	379
2000-4302	**A Review of Attitude Kinematics for Aircraft Flight Simulation** *W. Phillips, C. Hailey, and G. Gebert*	387
2000-4303	**A Generic Landing Gear Dynamics Model for LaSRS++** *W. Ragsdale*	412
2000-4304	**An Aerodynamic Model For Simulation of Close Formation Flight** *W. Blake*	421
2000-4305	**Neural-Network-Based Modeling of Rotocraft Vibration for Real-Time Applications** *S. Kottapalli*	432

TABLE OF CONTENTS

14-17 August 2000

AIAA Modeling and Simulation Technologies Conference and Exhibiton

AIAA PAPER NUMBER	TITLE *AUTHOR(S)*	
2000-4307	**An Advanced On-Board Airdrop Planner to Facilitate Precision Payload Delivery** *P. Hattis, T. Fill, D. Rubinstein, R. Wright, and R. Benney*	448
2000-4308	**Parachute Retraction Soft-Landing Systems using Pneumatic Muscle Actuators** *G. Brown, R. Haggard, and R. Benney*	459
2000-4309	**Performance, Control, and Simulation of the Affordable Guided Airdrop System** *S. Dellicker, R. Benney, S. Patel, T. Williams, C. Hewgley, O. Yakimenko, R. Howard, and I. Kaminer*	468
2000-4310	**Fluid-Structure Interaction Modeling of the US Army Personnel Parachute System** *R. Benney, K. Stein, T. Tezduyar, M. Accorsi, W. Zhang, and J. Leonard*	479
2000-4311	**Updated Flight Performance and Aerodynamics from a Large Scale Parafoil Test Program** *C. Madsen and C. Cerimele*	486
2000-4312	**Design and Testing of the X-38 Spacecraft Primary Parafoil** *R. Machin*	497
2000-4388	**An Object-Oriented Design For Trim** *J. Neuhaus*	507
2000-4390	**Managing the Playback and Recording of Simulation Models in an Object-Oriented Simulation** *P. Kenney and P. Sugden*	518

TABLE OF CONTENTS

14-17 August 2000

AIAA Modeling and Simulation Technologies Conference and Exhibiton

AIAA PAPER NUMBER	TITLE / *AUTHOR(S)*	
2000-4391	**An Object-Oriented Interface for Simulink Models** *M. Madden*	527
2000-4392	**Dynamic-Window Search for Unevenly-Spaced Parameter Tables** *S. Yoon, H. Kang, and C. Kim*	538
2000-4393	**Implementing Dynamic System Models in the ASSET Simulation Framework** *S. Derry and J. Maddalon*	546
2000-4394	**An Experimental Method for Estimating Dynamic Errors in Real-time Simulation** *R. Howe*	555
2000-4395	**Vectorization Blocks for Simulink** *G. Campa and M. Innocenti*	569
2000-4396	**Using High Level Architecture (HLA) for Distributed Simulation of Stochastic Petri Nets** *K. Bayarou and H. Szczerbicka*	575
2000-4397	**Fidelity Considerations for Civil Aviation Distributed Simulations** *Z. Roza, P. van Gool, and J. Voogd*	586
2000-4399	**Simulating Free Flight In HLA Federations** *Z. Roza and P. van Gool*	597

American Institute of Aeronautics and Astronautics

TABLE OF CONTENTS

14-17 August 2000

AIAA Modeling and Simulation Technologies Conference and Exhibiton

AIAA PAPER NUMBER	TITLE *AUTHOR(S)*	
2000-4400	**Real World Environment Interfaces for Modeling and Simulation** *W. Bezdek, T. Halley, and J. Samocha*	608
2000-4401	**An HLA Based Flight Simulation Architecture** *W. Huiskamp, H. Janssen, and H. Jense*	619
2000-4402	**High Level Architecture (HLA) Gateway/Native/Middleware implementations** *E. Hougland and D. Patterson*	627
2000-4487	**Joint Strike Fighter Virtual Air-to-Air Combat Analysis for the Joint Operational Requirements Document** *S. Bishop, D. Mittlesteadt, and T. Menke*	637
2000-4488	**Using Virtual Simulation for Tactics Development for Constructive Simulation** *S. Thomas and C. Martin*	646
2000-4490	**Air-to-Surface Virtual Simulation in Support of the Joint Strike Fighter's Joint Operational Requirements Document** *T. Menke, S. Bishop, S. Rodgers, G. Hellmann, and D. Mittlesteadt*	656
2000-4491	**JIMM: The Next Step for Mission Level Simulation Models** *J. Nalepka*	666
2000-4492	**Man-In-The-Loop/Hardware-In-The-Loop Synthetic Battlespace Simulation** *R. Perry, J. McDermott, M. Speed, W. Austin, and M. Foster*	674

TABLE OF CONTENTS

14-17 August 2000

AIAA Modeling and Simulation Technologies Conference and Exhibiton

AIAA PAPER NUMBER	TITLE / AUTHOR(S)	
2000-4495	**High Bandwidth Scene Projector Drive Electronics for IR Emitter Arrays** *M. Price*	682
2000-4496	**Structural Design Issues of Motion-Based Simulators** *M. Pagnotta, J. Callahan, and A. DiEdwardo*	693
2000-4499	**Simulating Aircraft Subsystems using Non-Homogeneous Automata Networks** *R. Hess*	700
2000-4500	**Data Management in the ASSET Simulation Framework** *J. Maddalon and S. Derry*	710
2000-4501	**Software Architecture for a Reconfigurable Flight Simulator** *C. Ippolito and A. Pritchett*	718
2000-4503	**DUECA - Data-Driven Activation in Distributed Real-Time Computation** *M. van Paassen, O. Stroosma, and J. Delatour*	728
2000-4504	**Rapid Prototyping of EFIS Software - Beyond Formats and Autocode** *E. Theunissen and R. Rademaker*	735
2000-4574	**Development of a New In-Flight Simulator MuPAL-alpha** *K. Masui and Y. Tsukano*	742

TABLE OF CONTENTS

14-17 August 2000

AIAA Modeling and Simulation Technologies Conference and Exhibiton

AIAA PAPER NUMBER	TITLE *AUTHOR(S)*	
2000-4575	**ACT/FHS - An Airborne Rotorcraft Simulator for Technology Development and Research** *C. Ockier and U. Butter*	753
2000-4576	**In-Flight Simulation as Tool for the Design of Future Pilot Assistance Systems** *P. Hecker*	764
2000-4579	**Flight Simulation using Actual Helicopter Equipped with Full-Authority FBW System** *S. Ozeki, K. Kimura, and T. Tomio*	774
2000-4580	**Recent Experience and Trends In United States Air Force In-Flight Simulation** *N. Weingarten and S. Markman*	783
2000-4581	**Improved Process for Validation of EFCS** *A. Persson and F. Karlsson*	794
2000-4582	**Model Development of a Level D Black Hawk Flight Simulator** *P. van Esbroeck and N. Giannias*	798
2000-4584	**Application of Large-Angle Data for Flight Simulation** *E. Dickes, J. Ralston, and K. Lawson*	809
2000-4585	**The Eleven Steps in Validation, Verification, and Accreditation (VV&A)** *K. Stengel*	822

TABLE OF CONTENTS

14-17 August 2000

AIAA Modeling and Simulation Technologies Conference and Exhibiton

AIAA PAPER NUMBER	TITLE *AUTHOR(S)*	
2000-4586	**Flight Data Reconstruction and Simulation of the 1992 Amsterdam Bijlmermeer Airplane Accident** *M. Smaili and J. Mulder*	833
2000-4587	**FBSS-RAIS Flight Crew Behavior Simulation System - Reconstruction of Accident/Incident Scenario** *K. Muraoka and N. Okada*	852

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37215 AIAA-2000-3974

THE USE OF FULL FLIGHT SIMULATORS & SYNTHETIC TRAINING DEVICES IN THE FIELDS OF HELICOPTER TRAINING, TESTING & CHECKING

Brian Hampson. F.R.Ae.S., AF.C.A.S.I.
President, Galileo Training Centre
Eli-Fly SpA. Esine (BS) Italy.

ABSTRACT.

The use of synthetic training devices have become accepted as the main tools for the training, testing and checking of aircrew of fixed wing aircraft. This has led to the adoption of standards both for the devices and the training for which they may be used and may lead to the removal the necessity of multiple regulatory qualification evaluations. Thus in the past quarter century there have been improvements in standards leading to a reduction in real costs. The same progress cannot be found in the commercial helicopter field. Why this difference in approach, especially as the military have embraced advanced simulation and has proved it works? This paper reviews some reasons behind this contradiction and attempts to explore the strengths and weaknesses of the technicalities of rotary wing simulators and the difficulties of training without access to STD. It traces the development of regulations that might change the situation. Finally the Author attempts to forecast where the present trends may lead and the difficulties still to be faced. To do this he uses his past experience as a trainer, as a member of a simulator manufacturing company and of the regulatory authority working groups that have produced new draft legislation.

INTRODUCTION

There are several military training centres both in the US and Europe in each of which there are more high-level helicopter simulators than there are commercial helicopter simulators in the rest of the world. There is just a handful of the latter and none of them is capable of providing the full range of training necessary for a commercial helicopter pilot. Why is this when for more than 25 years fixed wing simulators (FFS) and other Synthetic Training Devices (STD) have formed the foundation of the training for commercial fixed wing training, testing and checking? The larger fixed wing operators would not consider using their aircraft for training and checking today. During recent years the commuter and regional fixed wing operators have, increasingly, availed themselves of the same type of facilities. Military rotary wing training is likewise firmly based upon the use of the STD but few commercial helicopter operators have adopted the same practice.

I would like to examine the possible reasons for this situation and to see if we can draw some conclusions from the examination, that may cast light upon the potential for the future. To do this I will begin by reviewing the factors that have led the fixed wing commercial operators to choose the STD as their main training vehicle and then see what "read across" there might be to commercial helicopter operators. In my view the trend in the fixed wing world was predicated upon five different factors:

1. Regulations
2. Cost
3. Technology
4. Training Practices
5. Availability.

REGULATIONS.

Regulatory rules have existed in most countries for more than 50 years. These have permitted the use of some form of STD to be used in lieu of the aircraft for training, testing and for checking of competency of fixed wing pilots. In general, the stance has been that, the aircraft is the primary means of providing such services but that, with the prior permission of the Regulatory Authority, an approved synthetic device may be substituted for all or part of them. Commencing with instrument flight training and the ubiquitous Link Trainer we have now progressed to the use of Level D FFS that will permit all training testing and checking without any access to the aircraft at all. We are now on the threshold of seeing regulations being introduced that will mandate training on the STD instead of the aircraft.

In 1990 the FAA and NASA convened a workshop [1] of interested parties in Long Beach to debate the use

Copyright © 2000 by the American Institute of Aeronautics & Astronautics Inc. All rights reserved

of STDs in helicopter training. The proceedings of this Conference were later used as the basis of a text book on vertical flight training.[5] The results were not encouraging with even the simulator manufacturers doubting that there was an effective market for their products. One speaker estimated a worldwide sale of less than ten helicopter simulators during the decade of the nineties. Ignoring the sales of military simulators, that estimate has proved optimistic. What led the delegates to this somewhat pessimistic view?

COST.

The average helicopter costs a fraction of the price of any large passenger carrying fixed wing aircraft. Whilst a STD costing only 10% of the fixed wing aircraft was a good deal even if that 10% represented US$10 million such a device would represent the cost of several actual rotary wing aircraft. Moreover few helicopter operators had fleets of the size of the large commercial fixed wing operators. Even the largest helicopter operators who might have fleets of 100 plus aircraft had fleets of several different types whereas the major airlines would have 50 or more aircraft of the same type. Added to this, the large helicopter operators tended to have dispersed fleets where operations were based in several different parts of the world, usually where oil operations or explorations were centered. The commercial fixed wing operators tended to fly out of hubs from where all aircraft of a similar type were operated and maintained, providing a base of operations for their crews and the crew's training. It is also an unfortunate fact that helicopter STDs tend to be more expensive to purchase than fixed wing ones. This arises from several factors. Firstly, the dynamics of the rotary wing are more complex than are those of the fixed wing, meaning they require more software engineering and computer power. Additionally, tortional dynamics and vibration are of greater significance in helicopter training and operation and need to be represented well. To some extent this is offset by the reduced system simulation required in most helicopters but, as helicopters become more complex and greater advantage is taken of area navigation and auto flight systems this advantage will be reduced. An inspection of modern military helicopter cockpits gives ample evidence of this fact. Another factor is the visual system requirements for the helicopter. If it is to be used for close to the ground operations, the requirements are much higher than those for a fixed wing aircraft. The cost for the visual system then becomes a major part of the cost of the whole STD sometimes by a factor of two or more when compared to the small fixed wing aircraft.

Finally, there is the cost of data. Since the advent of the FAA's advanced simulation plan in the 1970's the data requirements for compliance with the regulations have been well defined for the fixed wing aircraft. Aircraft manufacturers and their equipment vendors are accustomed to these requirements and make provision for them in their aircraft development programs. Whilst not inexpensive, even so, there is data available and it is commercially priced. A typical simulator data package to meet the ZFT requirements can easily exceed US $ 2,000,000 for a large fixed wing aircraft. Many currently operating helicopters were designed without any recognition of the potential need for simulator data and even new helicopters are produced without the necessary simulator data packages being offered. The result is that the simulator manufacturer has to contract specialist data gathering companies to obtain the data for them. As this cost is unable to be amortized over a potential production run of several simulators it can prove an expensive exercise on even a simple helicopter.

TECHNOLOGY.

As already stated, the task of simulating a helicopter is more difficult than simulating a fixed wing aircraft. This is particularly the case in three areas. The first are the rotors. Until relatively recently, the computational capacity and speed of available computers was insufficient to properly reproduce all the dynamics of the blades, so the rotor was modeled as a disk. Faster and larger computers now permit the rotors to be represented as complex blade elements. Typically, modern rotor simulations are now computed by calculating the aerodynamic effects at six or seven stations along the individual blade's length with the result that the overall effect is much closer to the real rotor for any given set of conditions. This, of course, is more accurate but is more expensive.

As any helicopter pilot knows, vibration is a cue that is very important in flying many helicopters. Various methods of providing these cues over the necessary wide range of frequencies have been used. The traditional motion system developed for maneuvering cues and disturbance cues of fixed wing aircraft is not ideal for the helicopter. The sustained high frequency vibration typically found in helicopters is beyond the capacity of the six axis synergistic motion systems. It has to be supplemented by vibration platforms and vibration seats. Unfortunately both of these methods, whilst providing good vibration cues also introduce other unwanted problems to the helicopter simulator.

The vibration seat induces the vibrations to the pilot but, of course, unless supplemented by other means, the cockpit and particularly the instrument panels do not vibrate at the same time or the same frequency, as they do in the aircraft. To overcome this, vibration platforms on which the whole cockpit is mounted have been developed. These platforms are mounted on the motion system so that the other motion cues can still be produced. Again, this is an added cost. In addition shaking the whole of the simulator has bad effects upon the visual system which, as in the real world, has to be isolated from the simulator platform so that it is not prone to the induced vibrations.

Finally the greatest technical challenge is the visual system. Firstly, the field of view from most helicopters is much larger than that from a fixed wing aircraft. Many helicopters are, essentially, glass bubbles attached to rotors, engines and transmissions. Further the field of view is used much more than on fixed wing aircraft. Frequently the pilot uses the view from chin windows down by his feet to assist in landing, hovering and in some operational tasks. By the same token he likes to see the path of the rotor disk through the upper portions of the cockpit windows. Ideally a visual field of horizontal view of 200^0 x a vertical view of 100^0 required. Current mirror manufacturing technology cannot provide such a range and, even if this becomes possible in the future, the necessity of mounting the mirror in its optimum optical position is precluded because it will foul the structure of the flight deck and the visual display units. To all intents and purposes the collimated visual system will be limited to a vertical field of view of not much more than 57^0 and this does not even meet the current regulatory requirement for an FAA level D simulator.

TRAINING PRACTICES.

My original experience of helicopters was grounded in those in use in the military of almost 50 years ago. I then moved to fixed wing aircraft in the commercial aviation world and became immersed in the training regulations and practices appropriate to them. These were the drivers for the accelerating use of STDs. Major airlines faced with operating large aircraft fleets and crew complements of three or four persons and crewing ratios of six or eight crews per aircraft had two major problems. The need is for type training courses of perhaps 35 – 40 hours duration per crew. To this is added a requirement for each crew to be checked for competency twice a year as well as providing practice details and recency flying. It all culminates in the major operators having to contemplate many thousands of hours of flying training every year. With the costs of flying the aircraft for this purpose reaching figures of US$ 10,000 per hour or more the fiscal rationale for spending several millions on a sophisticated FFS are readily apparent. The same did not and still does not apply to the smaller helicopter operators and, in this sense they are similar to the regional airlines' position of five to ten years ago. The result is that the lure of the actual aircraft being used for training is very attractive. This is especially so because the utilization of the helicopter is frequently quite low. Thus there are always spare hours that can be applied to meet the training needs. If the aircraft's direct operating costs are reasonable, as they frequently are, it might be difficult to justify expending more money on any form of synthetic training device (STD). Another factor is that even those operators with large fleets of helicopters often have them dispersed all around the world so it does not appear to be attractive to set up your own training centre and equip it with sophisticated STD. To do so would mean the necessity of transporting the crew perhaps thousands of miles, with a consequent loss of utilization and with the addition of high travel and accommodation costs. Frequently these "aggravation" costs amount to more than the costs of operating the STD. However, it is my belief that it is for other reasons that STD are not used more widely in helicopter training. It is more to do with the currently common training practices in the helicopter industry. I was very surprised upon my return to helicopter operations to see the wide disparity between the fixed and rotary wing norms for training operations.

Examination of the current legislation in place in most countries, in respect of the training and licensing of helicopter pilots, shows a great deal of commonality and reflects the roots of the legislation. This should not surprise because most countries, as signatories of ICAO, have agreed to adopt the ICAO Annexes 1 and 6 as their minimum standards. These minimum standards have been used for many years, with only minimal conceptual change, primarily for the training and licensing of fixed wing pilots. When it came time to adopt similar standards for helicopter pilots it was natural and convenient to modify the fixed wing standards. In essence, these standards address the training, testing and competency of pilots in the normal, abnormal and emergency operating procedures appropriate to their type of aircraft. I have found, within the helicopter community, a great resentment at how often their operations have been

and still are assumed to be able to be legislated as if they were fixed wing operations.

What was missed in the adoption of this policy of standardization was that the scope of training, testing and competency checking leading to the issuance and renewal of a fixed wing pilot's licence was the for type of operation in which the aircraft was engaged. For the fixed wing aircraft, it is simple. They are primarily used in the carriage of passengers and cargo from point A to point B. The skills learned in basic training, modified only by type training during conversion, are generally adequate for the operation of the carriage of passengers and cargo. Only in a relatively small number of cases are additional fixed wing pilot skills required. These might include activities such as amphibious operations. To cater for these a separate rating is usually required for which there is added training and additional testing that needs to be periodically repeated in order to maintain the validity of the licence. The result is that upon gaining a commercial pilot licence, the newly qualified pilot can join an operator and after company specific and type specific training and testing, he can function fully in the commercial category gaining further experience "on the job". His periodic competency tests do not require him to carry out any operational drill nor demonstrate any skill that was not included in his basic and type training. In other words, the basic, and type training was appropriate to the type of operation as well as the type of aircraft.

No similar parallel situation exists in the rotary wing world's legislation. The student helicopter pilot, like his fixed wing counterpart, is trained to fly the aircraft and cope with emergencies and abnormal situations appropriate to the aircraft and that might reasonably be expected to occur in the flight of a helicopter moving from A to B. This is to say he can start up, taxi, hover, navigate, cope with system failures of various types and eventually land. As helicopters become more sophisticated, more pilots are expected to be able to fly in IFR conditions and systems that are more advanced become available to him. These are incorporated in the I/F and type training the helicopter pilot undertakes for which he receives an additional, annually renewable rating following the prescribed training. Up to this point there is no fundamental difference, in principal, with his fixed wing colleagues. Unfortunately, a helicopter pilot, with only these qualifications is virtually unemployable because only a very small percentage of helicopters are engaged in the carriage of passengers from A to B. The value of the helicopter lies in the activities that are specifically suited to its design and operation. Most commercial operators of helicopters are engaged in activities such as external load lifting, medevac, (HEMS) air/sea rescue, mountain rescue, oil exploration and drilling support, geological surveying and aerial surveillance of various types with security forces. None of these activities is covered in basic flying training nor are they directly reflected in the licensing requirements and the competency testing regulations. At best, some countries' regulations specify that there should be additional "experience" requirements to be met before engaging in such operations. In most cases, however, the absence of a legislated regulation is replaced by requirements laid upon operators by the companies that hire their services. During the contract negotiations, especially those that are most competitive, the potential employer seeks to ascertain that any company he hires for the work has a satisfactory record in the same business. This being taken as indicating competency in that field.

The fact that contractors and insurance companies lay these requirements upon the operators is both an indication of the need for such training and testing and a condemnation of the regulatory authorities for not having introduced the requirement sooner. The problem with the existing system is that it is neither comprehensive nor standardized. The levels of experience, required by the contractor, are varied by their perceived importance of the job and by the availability of the particular skills in the market at the time of contract award. The standards are, therefore, not standards at all. More importantly, the experience requirements are not so comprehensive as to demand that the abnormal situations and emergencies that can arise in any aircraft are demonstrated in the operational regimes in which the helicopter is flying. Thus, a pilot receiving "experience" in, for example, heavy external lift operations may not receive any experience of an engine failure or a tail rotor failure, in those conditions. The pilot gaining experience in mountain rescue will not gain practice in coping with an unexpected loss of visibility. Medevac experience will not include loss of power in situations of landing on or between high buildings in a built up area. Many other similar examples can be imagined. The mere fact that a pilot has hundreds or thousands of hours of operation in one of these categories does not mean that he has ever experienced a major problem during that time and, should he do so in the future he may be totally unprepared. This is not permitted in the fixed wing world. There, the reliability of modern aero-engines is now such that many commercial fixed wing pilots today operate for years without ever experiencing an engine failure at any stage of flight

let alone one in a critical phase of flight such as close to Vmca or Vmcg. In spite of this, all basic, type and recurrent training regulations include these exercises in the syllabi. If this is a requirement for the fixed wing pilot is not the same argument valid for the rotary wing pilot? "Experience hours" have some value but should not be confused with the necessary training.

There was some justification for these "oversights" in the past. Had the regulations required such training to be included and for competency to be demonstrated on a recurrent basis the operators would have faced difficulty in providing it. No commercial training devices capable of producing this level of training existed for some of the reasons discussed previously. The risks to aircraft and crew of carrying out the exercises in the helicopter itself were too great to accept. It is unrealistic to expect that all operators would be in a position to purchase STD of acceptable quality for their own individual use. A combination of all these factors has resulted in very lax, and I would claim, inadequate regulations being adopted for helicopter training in their specific areas of operation.

SOLUTIONS IN THE FUTURE

If I have been correct in identifying the underlying causes of the current lack of use of STD for helicopters are there any solutions and will they be adopted in the future? Each of my headings in this paper is related. None of them can be considered in isolation and any one will impact all the others. Improved technology impacts cost but also offers new training opportunities. Better simulation derives from improved technology and presents opportunities for changes in training practices. For training practices to be improved and standards raised will require new, permissive but not restrictive, regulations. All of these factors will improve the business case for the use of simulation in helicopter training. This will generate opportunities for more simulators to be built with a potential for reducing costs both of the STD and the training. Improved training can result in fewer accidents and incidents and in more highly skilled pilots. This, in turn, can reduce insurance premiums as well as operating costs. Let me now express my views of what can be done.

REGULATIONS FOR THE FUTURE

What is not generally recognized is that, during the same period that STD were being promoted by the FAA, in North America, for zero flight time training of fixed wing aircrew the use of STDs was prohibited, by regulation, for any tasks in helicopter operations. Sure, a few helicopter simulators were approved for some training but this was allowed by exemption. In the main, helicopter operators were being forced to continue to use their aircraft. In Europe, a similar situation prevailed with only three or four simulators being available to helicopter operators. These were only permitted to substitute for a few hours in the whole training program. In the future, I think that the first change must come from the helicopter Regulators. They must give the lead in agreeing to the criteria for the qualification of helicopter STD and promote greater use of STD in training. The European JAA is taking a lead in this by writing specifications for three types of helicopter STD.[3] There are the Full Flight Simulators[4] that, like their fixed wing counterparts, range from the rather simple Level A to the zero flight time capable Level D. Next in the hierarchy, and showing the greatest level of attractiveness it would seem, are the Flight & Navigation Procedure Trainers (FNPT)[5]. There are three levels of FNPT. Finally there are Flight Training Devices (FTD)[6] of which two levels are specified.

The FFS are specified on similar lines to those adopted by the FAA in AC 120-63. There are two major variances though. First, the JAA believe that the benefits of a collimated visual system in a fixed wing FFS outweigh the disadvantages of collimation. The same is not necessarily true for the helicopter FFS. Here the current and likely future limitation of mirror technology to one permitting a maximum of about a 60^0 vertical field of view makes this solution only marginally acceptable. A projected display, whilst not collimated, allows for much wider and higher fields of view. This permits the helicopter pilot to see the ground through chin windows as well as instantaneously seeing the rotor disk. For some types of operation and some helicopter configurations, the larger vertical field of view is critical. The trade-off, of course, is that only one pilot can get an optimally correct view of the ground without collimation so a second pilot on the flight deck will see an offset on an approach to a landing. This may not have a great deal of significance in helicopters where most are single pilot operations and runways are not frequently used. The degree of offset is a function of the distance between the eyepoints of the two pilots and in some helicopters this can be quite small leading to only minimal offset. Another solution is to permit a visual realignment to be selected from the instructors' station so that at any

time the handling pilot in a two pilot crew receives the correct view. It must be said, however, that this is not always possible in a dome and could be quite expensive to achieve.

Recognizing all these issues, the JAA standard will permit either a collimated image or a dome solution and the trainer is left to make the decision that he thinks best meets his particular situation. This is a good example of what I mean by permissive rather than restrictive regulation.

The Flight & Navigation Procedures Trainer (FNPT) is designed to meet the requirements of the new entry pilot and will permit much lower training costs for the initial CPL, ATPL and Instrument Rating. The lowest level FNPT is aimed at providing basic training in a simple, generic, single engined helicopter. It can be fitted with a visual system, although this is not mandated, but there is no motion or vibration simulated. At this time, the credits to be awarded to such a device are still in debate within the JAA. Informed opinion within the training industry suggests that, regardless of credits, an FNPT I can be used to give experience in many flight manouevers, including those close to the ground. The next level is the FNPT II and this differs from the FNPT I in that it has to be a twin engined device as presently defined but this may be subject to change. In either case generic aeroplane design is still permitted. It can also be used for the Multi Crew training (MCC) that is now a required part of the curriculum for the issue of any JAA commercial pilot licence. The Level III is more complex in terms of simulation and is aimed at providing training for the special types of flying in which helicopters are engaged. I must stress here that these criteria and the accreditation that may be given to these devices are still under development within the JAA and may not be fully realized until late this year. They should be considered in the context of the new JAA training programs for commercial pilot licences that allow two routes towards the CPL / ATPL. Either that of an integrated course of about 12 months continuous training or the modular course where the student can opt for one module at a time allowing him to earn sufficient money between modules in order to pay for the next one. It must be stated, however, that the present thinking is that FNPT shall only be approved for initial training and that no type testing or handling credits will be given.

The final type of device being discussed within the JAA is the FTD. Like its fixed wing counterpart it has to be type specific but, again, does not require either motion or visual systems. There is some debate within the industry as to what value there is in a device that may be relatively expensive due to data requirements but that is able to be used for very few training events. Few helicopters now flying have complex systems as may be found in the fixed wing world and without motion and visual the number of training events that might be approved will be reduced. One possible solution that is being touted is for the FNPT III to be built using approved type data so that it can be qualified both as an FNPT III and an FTD 2. This would be a cost-effective way of providing ab initio, special duties and some type training in a single device and may be ideal for use in a training school. The FAA does not have regulations covering the qualification of either FNPT or FTD for helicopters (unless by exemption). I would hope that they might follow the JAA lead in the near future both in the qualification of them and in the training events for which they may be used.

FUTURE COST ISSUES.

Helicopter FFS will continue to be expensive, although the gap between their costs and those of comparative fixed wing FFS will narrow. To accelerate the closing of this gap the helicopter aircraft manufacturers will have to accept, either in order to increase their attractiveness or by legislation, that data packages are a necessity. It was a long time before the fixed wing manufacturers accepted this and even longer before their vendor suppliers came to the same conclusion, so I am not expecting any miracles. The option provided by the proposed JAA qualification and approval of less complex FNPT & FTD may have a far greater effect upon the training costs. Any device, qualified, approved or not, that reduces the time necessary to reach or prove competence on the actual aircraft is a cost-effective device. If, as I suggested earlier, the use of these types of STD are approved for a range of training events, particular to helicopter training then costs will be less than those incurring the use of the aircraft. It is of note that the operating costs of the helicopter are greater than those of a comparably priced / sized fixed wing aircraft. Not only is it less economical in fuel consumption but its maintenance and, most importantly its insurance costs are higher. A fairly small helicopter such as the AS 350 or the Bell 212 cost in excess of US$1000 per hour to operate. Even though the capital costs of the FFS are high it can be sold at about the same price per hour of training, including the instructor. In considering the larger and more complex helicopters, their hourly operating costs can be twice this figure so the FFS and, particularly the FNPT/FTD become very cost

effective, as they provide more extensive training with a higher utilization rate.

Naturally, the rate of utilization is an important factor. Helicopters are rarely operated for more than 6 hours a day, if that, whereas a modern full flight simulator should be available for in excess of 20 hours per day, if the customer base is there. With utilization of this order the operating cost of the helicopter simulator will be very competitive, even during the initial years when capital costs are being absorbed. A typical business plan will amortize the capital costs over about ten to twelve years after which the cost of ownership, including operation will be lower by about 75%. In order to obtain utilization of twenty hours per day it becomes almost axiomatic to say that a training centre must operate the simulator(s) - that is either a third party company or by an operator who actively sells spare capacity. The advent of third party independent training centres offering training and competency testing can be an important factor so long as they are positioned reasonably close to the operating centre of the helicopters. It is only recently that these opportunities are becoming available. New helicopter training centres are being established in both North America and Europe, following the lead of the military again, and these will offer opportunities for competitively priced and highly optimized training for any helicopter operator. The SAS Flight Academy offers Bell 412 Level C training. Flight Safety International has a similar capability in both continents. Thomson Training & Simulation have announced a joint venture operation with Eurocopter to provide training for the latter's aircraft in Marseilles and my own company Galileo Training Centre will soon offer training for Bell 412, A 109 and SA 350 helicopters in addition to providing fixed wing ab initio training.

FUTURE TRAINING PRACTICES.

This is the area where the greatest need for change exists and that will provoke the greatest controversies, I suspect. As indicated earlier in this presentation, it can be seen that training is lacking in many areas of helicopter operations and that the regulations are nowhere near as demanding as they are for the fixed wing operations. In Europe, this may be set to change. JAR-OPS 4 (General Aviation – Helicopters – including Airwork) is to be published for public comment after it has been the subject of review by a helicopter sub-committee. It is likely to require the JAR-FCL (Flight Crew Licensing) to specify additional training and testing of pilots in the areas of operation typical to helicopters. This will replace the simple experience requirements now generally in use. It is inconceivable that the change will be made in JAR-OPS 4 without similar requirements being included in JAR-OPS 3 (Commercial Air Transportation – Helicopters). The major difficulty in addressing such training and testing requirements will be that it might prove impossible for them to be carried out on the aircraft. In this respect there is another parallel with the fixed wing aircraft. For some years the FAA have been looking at a fundamental change to the training requirements and now the JAA are facing the same issues. This would be to change the wording of current rules from a version that states "training shall be carried out on the aircraft or with the prior approval of the Authority, on a synthetic device properly qualified by them". The new wording would have to be on the lines of "training shall be carried out on a qualified synthetic training device or, with the prior approval of the Authority, on an aircraft". Many large operators would find no problem with this as applied to fixed wing aircraft but it would significantly change the whole training scene for the smaller carriers and all helicopter operators. Availability of sufficient simulators for all of these operators and in close proximity to their centres of operations would be a pre-requisite for such a change in regulation. The simulator manufacturers and training organizations would love to have this type of legislation introduced and, if you have a strong enough stomach, it might be a good time to invest in their stock. Only the most biased and entrenched trainer would be able to oppose the change on grounds of safety and improved standards but it will certainly, at least in the short run, cost them money. Some savings may also be possible. Insurance companies have already indicated that they would be prepared to reduce premiums for companies and individuals that subscribed to the more advanced training and higher licence qualifications.

It is on this reasoning that Galileo Training Centre is being established in Crotone in the south of Italy. Here we are building a state of the art simulator base that will provide a Level D Full Flight Simulator capability for the Bell 412, the SA 350 and the A109. Following upon the experience of the military, we are housing these simulators in a dome that will enable us to train pilots from ab initio, through type conversion to special operations. Included in the latter will be firefighting, air/sea rescue, mountain rescue, medevac and sling load operations. The FFS will be capable of providing training not only in these operations but will also encompass training to deal

with abnormal and emergency drills in these situations. Together with the helicopter FFS will be Flight Training Devices for the same types. In addition to the helicopter training we shall provide Flight & Navigation Procedure Trainers for fixed wing aircraft to be utilized in a commercial pilot ab initio training supporting the new JAA modular and integrated training criteria. The building and the land on which it is situated have provisions for expansion. Our development program foresees up to eight full flight simulators with additional classrooms briefing rooms and FTD / FNPT being provided. The Galileo Training Centre will have a fleet of fixed wing aircraft and helicopters too. Situated adjacent to Sant'Anna airport in Crotone the Centre will include maintenance facilities in support of the fixed wing aircraft and the helicopters thus serving as a support base for the helicopters used in the Provincial Fire Fighting contracts. Further developments will use the facilities of the Centre to provide maintenance training and the classroom training modules that are part of the JAA's modular courses for pilot licences. A large Conference room will permit the Centre to host aviation conferences and seminars. The helicopter operation have already commenced and the training centre building is scheduled for occupation in the Fall of this year, at which time the first of the FNPT will be installed. This will permit Galileo Training Centre to apply to the JAA for approval as a Flight Training Organization (FTO). A later application will be made for Technical Training approval too. The US$ 50 million facility has been supported by grants from the European Union as well as from the Italian Government as part of a program to bring employment to the area and to introduce high technology industry. Private investors have provided the balance of the financing. Another part of the government-funding program is to develop Crotone as a holiday resort. Although Galileo Training is not directly engaged in this part of the activity it is seen to be complimentary to the training programs in that it will provide a very pleasant background for the students and trainees of the Centre. I can quote from personal experience, gained over the last thirty months, to the fine beaches, weather, food and wine. There is no need for aviation training not to take advantage of these peripheral benefits!!

CONCLUSIONS

Given the slow progress, over the past ten years, in respect of changes in Regulations I cannot claim that the new opportunities I have covered in the last part of this paper will be all be introduced in the very near future. Certainly, I believe they are both long overdue and will be an important factor in the well being of the helicopter industry in the years to come. The possible advent of more tilt rotor aircraft and the benefits that these and other rotary wing aircraft can bring to reducing the congestion at the major airports is a major factor. By taking over the role of the small fixed wing aircraft on routes of less than 200 miles and using their unique routing capabilities they certainly offer the possibility of growth in the number of passengers carried. In turn this will provide the opportunity for more simulator training for the larger fleet of pilots that will be needed.

REFERENCES

(1) NASA/FAA Helicopter Simulator Workshop. NASA CP-3156. DOT/FAA/RD-92/2

(2) Vertical Flight Training. Edited by William E. Larsen, Robert J. Randle Jr. & Lloyd N. Popish. NASA Reference Publication 1373. DOT/FAA/CT-94/83.

(3) Joint Airworthiness Authorities (JAA), Joint Aviation Requirements – Synthetic Training Devices (JAR-STD).

(4) JAR-STD 1H Full Flight Simulators

(5) JAR-STD 3H Flight & Navigation Procedure Trainers

(6) JAR-STD 2H Flight Training Devices

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

AIAA-2000-4073

A00-37216

G-CUEING FOR FIGHTER SIMULATION TRAINING*

Marc-Olivier Sabourin† and Vincent Sondermeyer‡
CAE, Montreal, Canada

Abstract

An operator can be regarded as a set of sensors with a feedback algorithm (reflexes and skills), rules, and a knowledge base that are used in stabilizing and controlling an aircraft in a given flight path. Motion cueing devices play a role in the training tool by enabling the trainee to develop proper control strategy (tuning the gains of the feedback algorithm), and to acquire parts of the required set of rules and knowledge base (by simulating representative workload, for example). The motion system and the g-seat both have relative advantages and disadvantages and the choice of one, the other, or both must be made on the basis of an optimal alignment between the training objectives and the capabilities of the training device.This paper will describe an optimized g-cueing seat design and compare its training potential with a standard Stewart motion platform by evaluating their respective effects on the pilot control loop, on the required workload and on their effectiveness in facilitating the training of rule-based and knowledge-based behaviors.

Introduction

The need for motion simulation was recognized at the beginning of the twentieth century with the arrival of fighter airplanes in the First World War. From the beginning of aircraft simulation as a means of training, motion simulation has been attempted. Even if the seriousness of the first few attempts could be argued (Antoinette trainer[1], figure 1), they demonstrate the need for some form of sensory feedback. Although money was invested in the development of motion systems at the time, resulting in very imaginative technology, no research was conducted on the benefit of such devices.

It was not before the beginning of the 1960s that serious research was conducted on the effects of motion

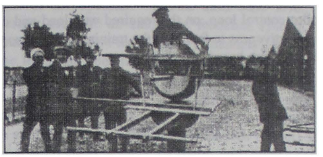


Figure1: The Antoinette Trainer

feedback on the pilot's ability to fly. Research conducted during this period demonstrated the need for sensory feedback in addition to visual feedback. Major results stated that 1: motion feedback helps the pilot reduce error in tracking tasks and 2: The vestibular system is responsible for an important part of the motion feedback. Since these findings were published, motion systems have been used and are now required by circulars such as the FAA 120 40C[2] for fixed wing aircraft and FAA 120 63[3] for rotary wing aircraft. Nowadays motion systems are a part of everyday life of simulation business. Yet, how a human perceives acceleration and most importantly the effect of training in a simulated motion environment is not well understood. For this reason, most users are still "in the

* Copyright © 2000 by CAE Electronics Ltd. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

† Group Leader, Military Flight Systems

‡ Motion Specialist, Military Flight Systems

dark" when it comes to selecting or specifying their need for motion simulation.

To complicate matters further, the Stewart platform is not the only product advocated as the best motion cueing solution on the market. Military customers often tend to select g-seats to provide motion feedback to the training [illegible] Cost, the requirement of a visual dome and high g-levels that cannot be reproduced by standard motion systems are among the primary factors which lead to the selection of a g-seat over a motion system.

While this paper will not try to eliminate doubts on the motion system selection criteria, it will describe an optimized g-cueing seat design and compare its training potential with a standard Stewart motion platform. This will be done by evaluating their respective effects on the pilot control loop, on the required workload and on their effectiveness to facilitate the training of rule-based and knowledge-based behaviors.

Human sensors

A human being subjected to accelerations will perceive movement from different elements of his/her sensory system. The human body can be regarded as a network of sensors interpreted by a central system, the brain. This network is comprised of several types of sensors. For the purpose of this paper we will distinguish the following 5 sensor groups.

1- Vestibular system: residing in the inner ear, the vestibular system can be compared to a three-DOF linear accelerometer coupled with a three-DOF angular velocity transducer. This sensor provides the reflex center with a feedback signal primarily used by the body equilibrium center and eye position control.

2- Visual perception of motion

3- Haptic system: sense of touch, which includes pressure gradients and sense of area of contact.

4- Proprioceptive: sensation of movement from internal organs.

5- Kinesthetic: Sense mediated by end organs located in muscles, tendons, and joints and stimulated by bodily movements and tensions.

Figure 2 illustrates the blend of different sensorial inputs with their transfer functions to the brain.

Each of the sensor types provides the brain with signals which, when combined, generate an acceleration feeling. However, a single stimulus of one sensor type can, even if incoherent with other sensors, cause the brain to conclude coherent motion with a visual system for example. While complete incoherence will normally result in motion sickness, a certain level of incoherence can pass unnoticed. In fact, coherence zones between motion and visual inputs were identified during a study

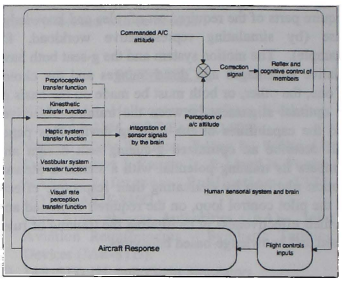


Figure 2: Aircraft man in the loop control system

conducted at Delft University and TNO Institute for Human Factors[4]. In these coherence zones, the angular velocity of the visual display could be changed while the motion angular rate remained the same without the subject noticing a difference. This is possible by the brain's ability to interpret signals from its sensors and to eliminate incoherent signals from the equation. It was shown that a Kalman filter emulates well the brain's ability to filter out incoherent sensorial signals, which are considered to be sensorial noise[5].

This action of the brain permits the use of different motion cueing devices that act solely on specific parts of the human sensors. When it comes to sensor

stimulation, both the six-DOF Stewart motion platform and g-seat exhibit limitations. In the case of a typical six-DOF Stewart motion system, all motion sensors will be stimulated simultaneously within the limitation of the system's operating bandwidth. In turn, this limits the action of the motion system on the proprioceptive sensor. Such is the case in the vertical axis when reproducing lower frequencies. Typical g-seat, on the other hand, will stimulate directly and artificially (without direct body motion) the haptic and kinesthetic systems, while providing only some cues through the proprioceptive and vestibular systems. In spite of their limitations, both devices will however provide motion cueing, which will feel coherent with the visual system motion perception.

Different sensor types will generate particular signals and will be interpreted differently by the brain. Stimulating one set of sensor or the other will likely not generate the same acceleration interpretation. Some sensors may have precedence when it comes to generating an automated feedback response or to judging an overall situation. The differences between the motion devices will thus be picked up, interpreted differently by the brain, and may generate different feedback responses.

Until now, most studies have been conducted on the effect of visual and vestibular system inputs as principal sources of motion feedback. It is now well know that the vestibular system is directly linked to the human reflexes in motion control. The role of the other sensors is more ambiguous. For certain simple situations, it is clear that the haptic system is involved in reflexes when linked with pain. However the interpretation of motion from haptic sensors is more complex than pain. The involvement of these systems in so many other sensations (other than motion) partly explains why they are less understood. These clues indicate that the sensorial weight assigned by the brain to the vestibular system and visual perception is higher than the one assigned to the other sensors for motion perception.

A recent study on acceleration perception depicts a more global sensorial picture. It demonstrates the possibility of segregating sensitivity of the different motion sensorial paths as a function of various parameters such as amplitude, frequency content and low level thresholds[6]. This suggests that the use of specialized devices targeted at stimulating specific sensors for specific acceleration characteristics (frequency, amplitude…) is possible. For example, the tactile feeling may be the most overwhelming sensation with high-amplitude, higher-frequency signals whereas for lower amplitudes, the vestibular system may be the principal sensor.

The control task and training

Based on studies conducted on the behavior of human operator[7], Rasmussen segregated the human behavior into three levels, based on the knowledge, rules and skills of the operator. At the skill base level, the operator acts like an automatic feedback system with inputs being its five main sensory channels. On the rule base level, the operator has learned to recognize signs (triggers such as an engine bird strike), then based on a procedure, the operator will initiate the required maneuvers. The maneuvers will then be executed by the skill base level using input directives (target) from the rule/knowledge base behavior and from sensory feedback for error correction. The knowledge base behavior is stimulated by cognitive understanding of the task and will require the operator to make conscious and logical decisions.

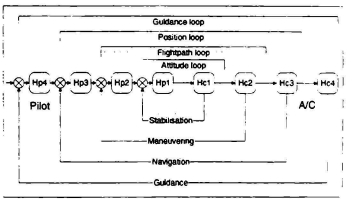


Figure 3: Aircraft man in the loop control system[9]

R. Hosman[8,9] illustrates the same principle on pilot control task using a nested control loop (figure 3). The inner control loop consists of attitude control and

involves skill based behavior. The outer loop consists of the knowledge base level guidance loop.

The goal in training pilots using flight simulators is essentially to develop and practice the pilots' skills, rules and knowledge. Different training goals will require different pilot control tasks (mission strategy, flight procedures, handling performance). In a relatively basic procedural trainer, only the outer control loop can be taught to pilots; all inner loops will have to be learned in the aircraft. As the simulation gets more and more accurate and representative of the flying environment, the training can be advanced to the more inner loops.

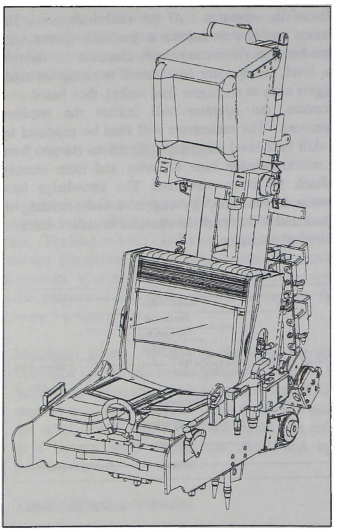


Figure 4: A dynamic seat for the BAe Hawk 115 Advanced Jet Trainer.

Training a pilot for the skill based stabilization loop is very similar to tuning controller feedback gains (figure 2). Of course, the pilot's skills consist of a complex controller, which will adapt to different control tasks. With increased time spent training, the pilot will adapt his control strategy to the aircraft response, behavior and limits. As the pilot interprets the aircraft response solely based on his perception of it, his control capability and the transferability of his training from the simulator to the aircraft will bemore dependant on the quality of the simulation of his sensorial feedback

The g-cueing seat

In 1998, NATO awarded CAE Electronics Ltd. (through Bombardier Aviation) a contract to provide four level-seven FTD simulators to provide training on the BAe Hawk 115 and the T-6A Harvard II (NFTC). Simulators were fitted with eight-channel visuals and eight-axis g-cueing systems.

To provide motion cues, T-6A and Hawk seats were modified to incorporate moving parts and inflatable sections. Figure 4 shows a schematic of the modified Hawk seat. The seats were designed to provide effective "onset" and "sustained-g" cues in four principal degrees of freedom: vertical, longitudinal, lateral and roll axis. Pitch and yaw cues are provided using combination of principal axis cues.

Table 1 lists all seat axes and cues for which they are used:

Axis	Cue
Seat pan pitch	Vertical and Longitudinal onset and sustain
Vertical seat back rest	Vertical and Longitudinal onset and sustain
Lateral seat back rest	Lateral onset and sustain and yaw onset
Rolling flapper	Roll, yaw and lateral onset and sustain
Lumbar cushion	Longitudinal sustain
Seat pan cushion	Vertical sustain
Lap belt	Vertical and longitudinal
G-suit	Vertical sustain

Table1: g-seat axis description.

Like most g-systems, this seat acts primarily on the haptic system by modulating skeletal attitude, muscle tonus, pressure gradients, and touch or area of contact as a function of the aircraft accelerations and rates. In addition it permits change in visual perception angle and acts on the vestibular system when providing onset

cues to the occupant. The distinctive characteristics of this seat are its ability to provide six-DOF onset and sustained cues while covering most of flight envelope and maximizing the sensorial coverage.

The seat was fitted with an advanced cueing model software controller. Divided into three channels, the software enables the direction of aircraft buffets, aircraft accelerations and seat micro-models to different seat axes. The software also controls the axis transfer function, buffet gains and micro-models' behavior. Modeling profiles are formatted using typical flight manual curves, which improves modeling efficiency and pilot interaction. Seat cue tuning was historically a complex challengeand required very long periods of simulator and pilot time. In this program, tuning is rendered most efficient through the use of an interactive graphical utility from which the engineer can subjectively tune on-line while the pilot is flying. This

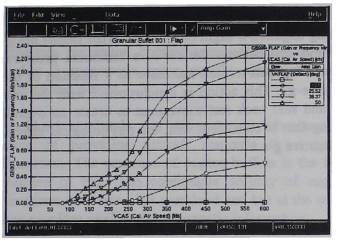


Figure 5: Flap buffet as a function of calibrated airspeed.

utility also enables rapid on-line modeling changes and addition of models, which enables extensive cueing experimentation in a short period of time. Figure 5 illustrates an example of the utility display for the amplitude gain of flap buffets as a function of air speed and flap setting.

Similarly, the transformation used for the transfer of flight accelerations to seat axis surface positions is controlled with the same graphical utility. Figure 6 displays the transfer function used for the flapper axis, to model sustained on-ground cues. Similar pages exist for all axes, onset and sustained cues, for both in-air and on- -ground cues. Extensive testing with experienced fighter pilots resulted in an optimal transfer function for each seat degree of freedom.

Principle of operation

G-seats differentiate themselves from other motion seats by the sensory path they emphasize and by how the various sensory paths are combined with each other. Another important aspect in the evaluation of g-seats is side effects that can be created while trying to stimulate a specific sensory path. For example, a device could generate an eye height change which would correspond to an upward acceleration and at the same time generate a gradient of pressure that would correspond to a downwards acceleration. In this case, the gradient of pressure is a side effect, and creates a confusing overall sensation.

Another type of g-system uses a network of inflatable air bellows to modulate seat surface hardness. Inflating the bellows harden the contact surface, giving the impression of increased g. In doing so, the occupant is lifted to a higher vertical position which causes the opposite sensation of reduced g.

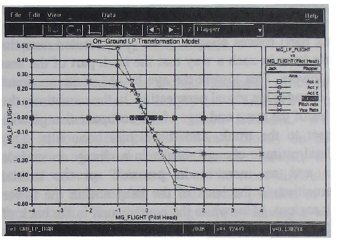


Figure 6: Flap buffet as a function of calibrated airspeed.

The proposed g-seat described in this study combines principles from moving surfaces and inflating bellows concepts to alleviate the negative effects of each cueing method, maximize the sensorial coverage and produce an overall improved sensation. The principle of operation of the seat is described in detail through the vertical axis as it covers the complete range of cueing methods used on the seat.

The heave cue

In earlier seat designs, heave (vertical) acceleration was simulated by vertical seat pan movement which also induces "back scrubbing" (i.e., change in tension change in the skin of the user's back) due to the back rest remaining vertically stationary. Therefore, the amount of back scrubbing generated is not controlled. Experiments conducted in prior studies conducted at CAE revealed that a pitch movement of the seat pan coordinated with a back rest vertical displacement generates cues on the haptic sensory system along with an eye height change resulting in realistic heave acceleration cues.

The main goal of the proposed g-seat vertical axis movement is to provide a motion simulator seat in which seat pan movement in pitch is coordinated with seat back movement in order to simulate sustained vertical acceleration with minimal side effects. The movement of the seat pan and seat back are controlled in response to the heave acceleration of the simulated fighter. The pitch movement of the seat pan stimulates several sensory paths in a manner consistent with the physiological effects caused by an acceleration. The part of the seat pan which is nearest to the backrest (back part) rotates about a pivot point located near the user's knee in such a way that the back part of the seat pan lowers. This lowers the eye height of the occupant and changes his visual perspective. In the meantime, the angles of the arms and legs of the occupant change with respect to the trunk, and the ischial bone tilts. This skeletal attitude change results in a change in muscle tone and an increase in pressure on the ischial tuberosities, in the back of the knee and in the lower back regions. Meanwhile the contact area in the back of the knee and lower back region is augmented.

A change in the seat pan pitch angle while the seat back remained stationary would naturally create back scrubbing, which might result in an unrealistic cue if its amplitude were either too big or too small. Therefore, the seat back adjustment is used to provide a controlled amount of back scrubbing. The moving seat back can also push and pull the occupant into and out of the seat pan, therefore enhancing the pressure gradient on the seat and on the lower back region. As explained above, a well-synchronized and properly-adjusted movement of the seat back and seat pan results in a cue that feels realistic and reproduces a fair number of the effects experienced by a pilot who is subjected to a real heave acceleration.

In addition to the seat and the back, three other channels act during heave cueing. With increasing positive g-levels, the g-suit inflates proportionally or according to the aircraft's specifications. The seat pan localized bellows inflate slightly, rendering the seat surface harder, providing enhanced tactile sensation of pressure. The balloning effect experienced on similar bellow systems is minimized using a minimal area hard covered bellow which limits deformation. The visual perception angle is controlled by modulating the seat pan pitch position. And finally the lap belt releases its tension, reinforcing the feeling of compression against the seat caused by g-forces. The above mentioned effects are all consistent with the physiological effects caused by an upward acceleration.

During negative-g flight (such as inverted flight) the g-suit and seat bellows deflate to a minimum pressure, and the lap belt tension increases as a function of the negative-g. This action is synchronized with the seat pan and seat back rest movement to create an overall feeling that is consistent with the physiological effects of negative g's.

Actual results

Pilots' comments in general are that the seat is their only indication of the aircraft performance without having to constantly revert to the instruments. Some areas where the need for g-cueing was expressed include on ground handling, high g maneuvers, and air-to-air refueling maneuvers. These maneuvers will be associated with the training type feedback discussed previously.

Improved realism aids the pilot to better concentrate on flying. When taxing along the ground the rumbling frequency gives an indication to the pilot of his ground speed and the condition of the runway. The moment that each wheel leaves the ground on liftoff is associated with a reduction in buffeting.

G-suits are, for all purposes, the only means of simulating high g maneuvers. As the pilot begins to feel the inflating of his suit, his intuition causes him to anticipate changes in aircraft handling. Here the pilot has adopted rule training to better anticipate changes. Buffets on the Harvard were found to be the only indication of a stall onset, as the aircraft is very stable aerodynamically. Without buffeting, the student would have no clear sign to with which to associate the onset .

Tracking performance is measured in terms of the pilot's ability to maintain an ideal aircraft position. In air-to-air refueling maneuvers, distances are often judged by buffeting caused by turbulence created by the re-fueling aircraft. A slight yaw cue is used by the pilot to confirm that contact has been made. This contact could not otherwise be detected. Another example of this the shifting of the back rest on the g-seat in response to a small change in lateral acceleration gives the pilot the onset cue that is needed for immediate correction. In general the seat surfaces were tuned to be very sensitive to relatively small accelerations providing maximum control sensitivity to the pilot. Higher g maneuvers were mostly simulated using bellows and g-suits.

Workload similitude can either be an advantage in terms of providing aircraft response or a tiring distraction. The presence of engine buffets is the best example of this. The continuous droning of engine vibrations increases pilot workload and but at the same time provides vital engine diagnostic information.

With the use of the g-system, the simulator becomes less forgiving. If the gears are not retracted following take off, or a heavy touchdown is conducted, the pilot is made aware immediately. These normally evident flying errors would otherwise pass unnoticed. The g-system's best trait however is its ability to provide the pilot with subtle cues alerting him to changes in aircraft attitude. This in turn aids tracking as pilots are able to take intuitive corrective action. Tracking tasks such as formation flight has been improved by using the seat as a cueing device.

Pilots have commented that more representative workload and acceleration feedback is critital in helping them to fly the simulator. The presence of buffeting increases the realism and provides critical information to the pilot for flight diagnostics.

Discussion

As shown by the results, the g-seat is beneficial if not essential for simulator training. The greatest benefits of the seat can be summarizedby two statements:

"The seat enables detection of events such as stall onset and touchdown that could not be detected otherwise."

"The seat provides the missing cues in such a way that the trainee can concentrate on the bigger picture of training instead of spending all efforts to control the simulated aircraft."

What the first statement says is that the seat is actually essential in providing rule based training as defined by Rasmussen[4]. Effectively, in this case, the seat becomes another source of information, like the instruments and the visual system, that cue the operator to execute a learned rule. The advantages of this feedback is clearly its similitude with the aircraft and the fact that it uses a separate sensory channel. The visual sensory path is often overwhelmed with all available information and less efficient at gathering additional critical information. The pilot will thus increase his efficiency by using alternate sensory channels to detect and interpret aircraft flying characteristics.

This is reinforced by the second statement, which says that some information is simply not available or is too difficult to derive from the available feedback. When flying in formation, pilots will rely mostly on their acceleration feedback to stabilize the aircraft at a constant velocity. This task would be more difficult to execute by only relying on the visual position of the other aircraft. The eight-axis g-seat, by having a wide sensory feedback coverage, actually brings the flight experience closer to the real world and improves pilot's ability to control the simulator.

However what has not been defined yet is whether or not the skill base behavior developed while training on a g-seat is directly transferable to the aircraft. The results of the tracking tests[11,12] and control experiments[10,11] did demonstrate that a g-system can improve a pilot's performance in the simulator, however, it is not clear how much of the pilot control loop 'tuning' will have to be adapted when flying the real aircraft after a training session. Clearly, the more consistently the g-seat covers the sensory feedback spectrum with the physiological effect of acceleration

perceived in the aircraft, the more the training will be transferable without adaptation.

Effectively, in most cases, a g-seat partially stimulates kinesthetic, haptic and proprioceptive systems. All g-seats will always have conceptual limitations (such as travel within the cockpit) and the extent at which they stimulate the vestibular and proprioceptive systems is limited.

When using eithera Stewart motion system or a g-seat, a tuning compromise must be made in order to achieve optimal usage of their limited travel. When using a motion system for a skill base training goal, a typical compromise made would be to favor the use of most of the travel for small accelerations while limiting the high-g maneuvers. This compromise will favor skill base training during low g level maneuvers, which corresponds to the most frequent flight regime. On the other hand, the use of this trade-off will cause high g-level maneuvers to lack the representative workload provided by the g-seat. In addition, the lower frequencies for the vertical axis and the mid-frequencies (~2 radians/second) for longitudinal and lateral axis will lack cues. A combined motion and g-seat solution with complimentary tuning would resolve these issues.

When the aim of the training is to maximize transferability of rule base and knowledge base behavior to the pilot, the described g-seat is the most efficient solution. However, if the goal of training includes the optimal transfer of skill based behavior, a combined motion/g-seat solution should be considered.

Summary & Conclusions

The g-seat coupled with the visual system and simulated cockpit maximizes transferability of rule based training.

The g-seat described in this paper facilitates training activity by providing otherwise unavailable flight condition information. The seat helps trainees to fly the simulator with more ease, thus enabling them to concentrate on other training objectives of the session, such as the mission.

When one of the training goals is to maximize transferability of skill based behavior, a combined motion/g-seat solution should be considered for training pilots on fighter aircraft. Such a goal may not be reached with only one of the two devices used independently because of their individual limitations.

The cueing control software developed for the NFTC program enabled extensive cueing experimentation and rapid convergence to optimal g-cueing.

The eight-axis g-seat provides improved sensorial coverage and improves handling quality and tracking performance while reproducing representative workload.

Acknowledgements

The authors would like to thank the following individuals for their contribution to this program: Chuk Ng, Eric Richard, Marc Kyriaco, Malcolm Lewis, Reginal Decoste, Dan McWilliams, Slobodan Tomovic, Joshua Todoros and Giovanni Minotti.

References

[1] J.M.Rolfe & K.J.Staples, Flight Simulation. ISBN. 0 521 35751 9 paperback.

[2] AC120-40C Airplane Simulator Qualification, US Department of Transportation, Federal Aviation Administration, 1998

[3] AC120-63 Airplane Simulator Qualification, US Department of Transportation, Federal Aviation Administration, 1994

[4] Steen, F.A.M van der (1998). Self-Motion Perception. Ph.D. Thesis. Delft University of Technology. Delft the Netherland, ISBN 90-9011757-1

[5] Borah, J. Young, L.R. and Curry, R.E. (1988) Optimal Estimator Model for Human Spacial Orientation. Annals New York Academy of Sciences. Volume 545 December 26 1988, Presented at the Symposium on Representation of Three-Dimensional Space in the Vestibual, Oculomotor, and Visual Systems June 3, 1987.

[6] Zaichik L.E., Rodchenko V.V. Rufov I.V., Yashin Y.P. and White A.D. (1999) Acceleration Perception, AIAA-99-4334

[7] Rasmussen, J. (1983). Skills, Rules, and Knowledge; Signals, Signs, ans Symbols, and Other Distinctions in Human Performance Models. IEEE Transctions on Systems, Man and Cybernetics, Vol. SMC-13 No.3, June 1983.

[8] Hosman, Ruud, (1999). Are Criteria for Motion Cueing and Time Delays Possible? AIAA Modeling and Simulation Technologies Conference Portland, OR, 9-11 August.

[9] Hosman, R.J.A.W (1996) Pilot's Perception and Control of Aircraft Motions. Ph.D. thesis, Delft University of Technology. Delft University Press, Delft, then Netherlands ISBNN 90-407-1384-7

[10] White, A.D. (1989) G-Seat Heave Motion Cueing for Improved Handling in Helicopter Simulator. simulators Presented at AIAA flight simulation technologies conference, boston MA. 89-3337-CP

[11] Martin, E.A., Osgood, R.K., McMillan, G.R., (1987) The Dynamic Seat as an Onset Cueing Device. AIAA 87-2438-CP

AIAA Modeling and Simulation
Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37217

AIAA 2000-4075

DEVELOPMENT OF A SUSTAINABLE-G DYNAMIC FLIGHT SIMULATOR*

C.H. Spenny† and B.S. Liebst‡
Air Force Institute of Technology
Wright-Patterson AFB, OH

T.L. Chelette
Air Force Research Laboratory
Wright Patterson AFB, OH

C. Folescu and J. Sigda
Environmental Tectonics Corporation
Southampton, PA

Abstract

This paper discusses two related design issues that are critical to the development of dynamic flight simulation (DFS) capability using a man-rated centrifuge: 1) response time of the centrifuge in an operator-controlled simulation of a high performance aircraft and 2) the effect on an occupant of angular acceleration artifacts produced by a centrifuge that provides rapid response. The ability of a centrifuge operated in the DFS mode to meet the response recommendations of the Federal Aviation Administration (FAA) for motion simulators is discussed and a model-based controller is shown to improve the response capability of some existing centrifuges sufficient to meet the FAA recommendations. Human response tests have been conducted on the centrifuge at Wright Patterson AFB to evaluate sensitivity to the artifacts produced by a centrifuge. Results indicate the effect to be no more than a mild disturbance over the expected range of g-loading and artifact magnitudes. The unique feature of this type of motion simulator is that the pilot can be exposed to high-fidelity, sustained, elevated g-levels while receiving training in flight procedures and air combat tactics. The expected payoff for DFS-based training is reduced loss of life and equipment while becoming more proficient at practicing the most demanding maneuvers under g-stress.

Introduction

Ground-based motion simulation of aircraft is currently accomplished with hexapod-type devices. These devices are able to provide high fidelity acceleration cues with suitably little addition to the response time a pilot observes in the actual aircraft being simulated. This makes them particularly suited to providing motion cues for aircraft operations at low g-levels such as landing tasks where pilot response is critically dependent on the fidelity of the visual and motion cues provided by the simulator. The device response time is predominantly determined by the inertia of the cockpit capsule that must be accelerated, the size of the actuation devices that produce the forces and the effectiveness of the controller in matching the desired input.[1]

Hexapod devices are not able to provide sustained acceleration levels due to stroke limitation of the

*This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

†Associate Professor of Aerospace and Systems Engineering, Member

‡Professor and Head, Department of Aeronautics and Astronautics, Member

[1] The presumption being that computational and transport delays in modeling aircraft flight are negligible by comparison, which is generally the case given the capability of current computer hardware.

prismatic actuation devices that provide the forces. This means that flight fidelity is diminished in many maneuvers such as a basic coordinated turn or critical tactical maneuvers of fighter/attack aircraft. Centrifuge-based flight simulation offers the potential to provide sustained high-g flight simulation. Whether a centrifuge might also provide high fidelity flight simulation for low g operation is an open issue that is addressed in this paper.

The most fundamental challenge in imbuing a centrifuge with high fidelity DFS capability is to provide rapid response with a massive device: 1) whose controlled inertia includes a planetary arm as well as the cockpit capsule (referred to in this paper as the cab) and 2) whose changes in acceleration level result in acceleration artifacts, namely angular accelerations associated with cab rotation, that may degrade the riders perception of flight. Failure to provide adequate response can result in pilot induced oscillations during cue-critical events, in simulator sickness and general degradation in transfer of training.

The class of centrifuge for which a model and test data has been developed to address these issues is a 3 degree-of-freedom, high onset-rate machine, the GFET (G Flight Environment Trainer) centrifuge built by Environmental Tectonics Corporation (ETC).

Definition of Dynamic Flight Simulation

Dynamic Flight simulation (DFS) is defined in this paper to be operation of a centrifuge as a flight motion simulator with the centrifuge driven by pilot-commands in response to a flight condition perceived in the device. It is similar to a hexapod motion simulator in that a pilot provides closed loop response to out-the-window visual cues, instrument readings and perceived motion cues that have been coordinated to represent aircraft flight. Relative timing of this sensory information to the pilot is critical and nearly as significant is the need for response time to be similar to that of the actual aircraft. The implications for ideal operation are that the math model of aircraft response must compute instantaneously and the centrifuge should respond instantaneously to the math model output.

There are recommendations in Federal Aviation Administration Circular 120-40B for how much motion and visual tracking delay can be tolerated in simulators used to train commercial pilots.[1] Table 1 summarizes the FAA recommendations for two classes of simulators. It must be noted that the simulators for which these specifications were written are hexapod devices that impart only onset acceleration cues that are "washed out" within a fraction of a second. Hence, it does not address the issue of how much lag is suitable for a pilot experiencing periods of sustained acceleration levels above 1g. It is likely that the pilot will be less sensitive to lag at elevated g-levels, but this is conjecture on the part of some of the authors that needs to be evaluated with human response tests.

Simulator Type	Response	Tolerance
A, B	Motion	$\leq$ 300 milliseconds
	Visual	$\leq$ 300 milliseconds $>$ motion lag
C,D	Motion	$\leq$ 150 milliseconds
	Visual	$\leq$ 150 milliseconds $>$ motion lag

Table 1: Recommendations of FAA Advisory Circular AC-120-40B for maximum tracking delay of motion-based simulators used in pilot training.

Figure 1 compares the tracking delay of two existing GFET centrifuges and one that is under construction. Tracking delay is defined for a DFS centrifuge to be the sum of the transport lag and the delay in centrifuge response, where transport lag is the sum of signal processing delays associated with passing the stick input to the centrifuge. Each of these machines can be operated in the DFS mode although that is not the principle mode of operation for any of them. A continual reduction in delay is noted, the result of design improvements, namely: increased motor size, reduction of centrifuge mass in motion, reduction of transport lag and increased speed of the computational hardware and software. The second and third meet the recommendations of FAA category A, B simulator but none currently meets the FAA more stringent category C, D recommendations. The Generation I, serial #1 GFET has been operated in the DFS mode and anecdotal response of riders indicate it has adequate fidelity. This paper presents an improved controller design that is expected to make any of these centrifuges capable of satisfying the category C, D recommendations.

The design philosophy that has been adopted in making the GFET into a DFS is to meet the recommendations of Circular 120-40B for operation of C, D type simulators when below 3g's, the range of normal aircraft operation where pilot's are most sensitive to fidelity of simulator motion. For operation above the 3g-level, increased power requirements is likely to saturate the control hardware. The next step in evolving DFS capability is to evaluate the

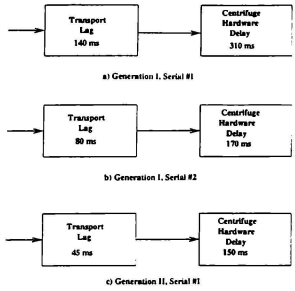


Figure 1: Comparison of the tracking delay of three GFET centrifuges with DFS capability.

performance of the hardware that meets the requirements below 3g for suitability above 3g's. Human response tests are necessary to make this evaluation. Response capability can be increased if these tests suggest such. It is prudent, however, to first evaluate the capability described in this paragraph since increases in motor size and reductions in centrifuge weight can have enormous impact on cost of centrifuge infrastructure and operation.

Model of DFS

A model of centrifuge dynamic response in the DFS mode has been developed in order to evaluate control algorithms and to predict pilot response prior to implementation. Figure 2 is a module representation of the model indicating feedback of centrifuge (plant) response to match input profiles of the desired accelerations from the aircraft simulation. In this paper, the focus is on the control module which will be discussed in some detail including the g-pointing architecture that is implemented in GFET machines and the form of the feedback control. Output data from the Output Processor module will be presented that indicates the level of DFS fidelity attainable and the magnitude of artifacts. Other modules are not relevant to the topic of this paper and are not discussed. The mass, structural and aerodynamic parameters of the Generation I, serial #1 GFET machine have been used for the data presented in this paper. The model has been validated sufficiently by response tests conducted on the GFET machines such that the model can be used confidently to assess the effect of design changes in centrifuge and

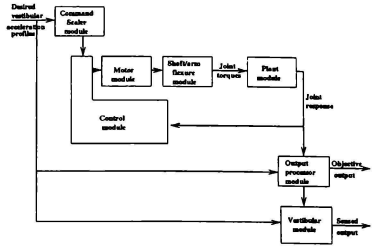


Figure 2: Module representation of the centrifuge math model.

pilot response. The model was implemented with MATLAB^{TM} and SIMULINK^{TM} software.[2]

G-pointing Control Strategy- With only three controlled degrees of freedom in the centrifuge, it is not possible to simultaneously match the six independent components of aircraft acceleration. Dynamic flight simulation, in one implementation, reproduces the three instantaneous components of aircraft rectilinear acceleration, synchronized with the pilot's stick, throttle and pedal inputs and with the instantaneous visual imagery of flight condition presented to the pilot. This control strategy has been referred to in the literature as 'precise g-force control'[3] and as 'g-pointing'[4]. These names derive because the cab occupied by the pilot is continuously and actively 'pointed' relative to the instantaneous acceleration vector of the centrifuge so that the three components of desired rectilinear acceleration in the aircraft (and vestibular) coordinate frame are obtained. In figure 3, the centrifuge acceleration vector, which is the resultant of normal and tangential acceleration of the planetary arm and gravity, and which varies with time, can be distributed to the three components of the vestibular frame acceleration ($\ddot{x}_v, \ddot{y}_v, \ddot{z}_v$) by proper selection of angles q_2 and q_3.

While the rectilinear accelerations mimic those of the aircraft as precisely as the controller is able, the rotations that must be imparted to the cab to realize this, actually produce angular acceleration artifacts that degrade angular acceleration fidelity. A primary concern is assessing the impact of these artifacts on pilot perception and in minimizing them.

[2]MATLAB and SIMULINK are registered trademarks of MathWorks, Inc.

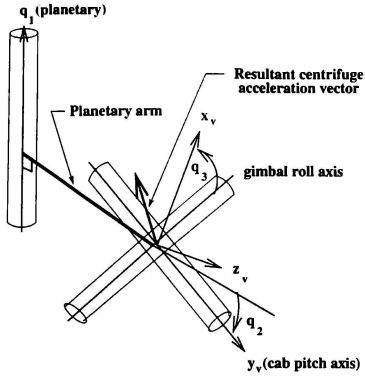


Figure 3: Kinematic structure of the GFET 3 degree-of-freedom centrifuge.

The quality of the g-pointing algorithm and the amount of artifact is determined quantitatively by the model that has been developed and is reported in this paper along with the results of human response tests that evaluate the effect of the angular acceleration artifacts.[3]

Implementation of a g-pointing controller implies partiality toward precision in the rectilinear acceleration at the expense of large angular artifacts. There is a pragmatic reason for this- namely, a centrifuge is able to deliver sustained g's but can only deliver the onset of angular cues, since it has angular limitations similar to the hexapod. Thus, if it could be shown that high angular acceleration artifacts were not disruptive to DFS, g-pointing may be an appropriate form of DFS control. To assess this conjecture, it is necessary to run subjective tests of human subjects under controlled conditions.

PID Control of the Cab- As indicated in figure 3, the gimbal roll axis and the cab pitch axis intersect at a point along the planetary arm and the origin of the vestibular frame (x_v, y_v, z_v) is at this intersection. The gimbal roll axis is perpendicular to the planetary arm (i.e. tangent to the arc created by the intersection point). The cab pitch axis is perpendicular to the roll axis and the y-axis of the vestibular frame lies along the pitch axis.

The three components of vestibular frame sensed rectilinear acceleration $(\ddot{x}_v, \ddot{y}_v, \ddot{z}_v)$ would lie along the vestibular frame axes depicted in figure 3. These components are related to the gimbal (q_2) and cab (q_3) joint angles as follows:

$$\ddot{x}_v = -L_1C_2S_3(\dot{q}_1)^2 - L_1C_3\ddot{q}_1 - S_2S_3g \quad (1)$$

$$\ddot{y}_v = L_1S_2(\dot{q}_1)^2 - C_2g \quad (2)$$

$$\ddot{z}_v = -L_1C_2C_3(\dot{q}_1)^2 + L_1S_3\ddot{q}_1 - C_3S_2g \quad (3)$$

where L_1 is the length of the planetary arm and S_i and C_i, are the sine and cosine, respectively, of angle q_i. To point the cab, it is necessary to obtain the joint angles q_2 and q_3 that correspond to the desired vestibular accelerations ($\ddot{x}_v^{des}$, $\ddot{y}_v^{des}$). The computations, as indicated in figure 4 by the two blocks labeled 'Solve for q_i', are readily obtained by solving equations 1 and 2 for q_2 and q_3, respectively:

$$q_i^{com} = \arcsin\{-b/2a \pm \sqrt{(b/2a)^2 - c/a}\} \quad (4)$$

where for $i = 2$, $a = L_1^2\dot{q}_1^4 + g^2$, $b = -2\ddot{y}_v^{des}L_1\dot{q}_1^2$ and $c = (\ddot{y}_v^{des})^2 - g^2$ and for $i = 3$, $a = (L_1\dot{q}_1^2\cos q_2 + g\sin q_2)^2 + L_1^2\ddot{q}_1^2$, $b = 2\ddot{x}_v^{des}(L_1\dot{q}_1^2\cos q_2 + g\sin q_2)$ and $c = (\ddot{x}_v^{des})^2 - L_1^2\ddot{q}_1^2$.

The three blocks labeled 'PID Compensator' in figure 4 are the typical proportional-plus-integral-plus-derivative controllers [5] similar to that used in the GFET machines. There are three gains in each channel used to tune the system for optimum performance. The centrifuge hardware delays indicated in figure 1 are obtained with this type of control.

The angular acceleration artifacts for the g-pointing strategy are given by:

$$\alpha_x = -C_2S_3\dot{q}_1\dot{q}_2 + (S_3\dot{q}_2 - C_3S_2\dot{q}_1)\dot{q}_3 - S_2S_3\ddot{q}_1 - C_3\ddot{q}_2 \quad (5)$$

$$\alpha_y = S_2\dot{q}_1\dot{q}_2 - C_2\ddot{q}_1 - \ddot{q}_3 \quad (6)$$

$$\alpha_z = -C_2C_3\dot{q}_1\dot{q}_2 + (C_3\dot{q}_2 - S_2S_3\dot{q}_1)\dot{q}_3 - S_2C_3\ddot{q}_1 + S_3\ddot{q}_2 \quad (7)$$

Sample Mission Profile- A pop-up maneuver defined analytically in reference [2] contains a correlated set of the six components of aircraft accelerations. The maneuver includes a pull-up from low altitude with thrust simultaneously applied, followed

[3] There are other control strategies that have been investigated and even adopted[3] that control the rectilinear and angular accelerations in a weighted fashion so as to limit errors in either from becoming too large. This paper identifies reasons why g-pointing might be preferred but does not reach a conclusion as to which strategy might be better. The results obtained here for the g-pointing strategy can also be applied to the weighted strategies.

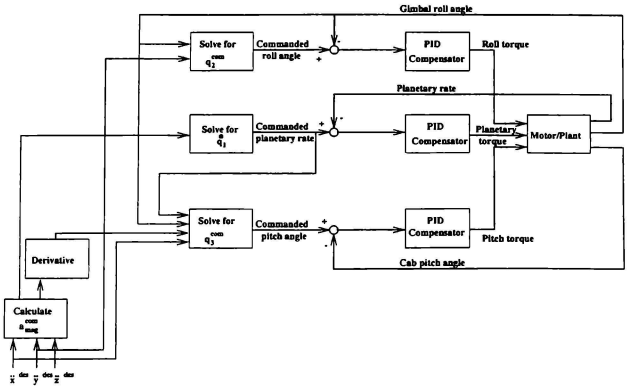


Figure 4: Block diagram of the G-pointing control architecture that uses PID control compensation.

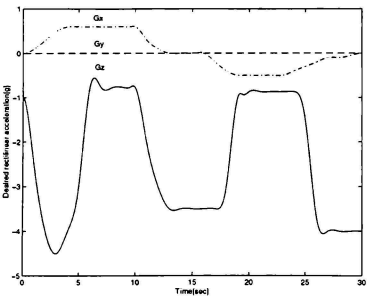


Figure 5: G_x, G_y, G_z accelerations for the Pop-up maneuver.

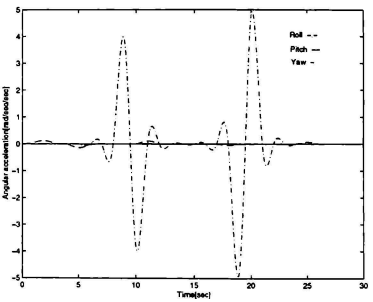


Figure 6: $\alpha_x, \alpha_y, \alpha_z$ accelerations for the Pop-up maneuver.

by a roll-over (180 degrees), followed by a pull-down. Thrust is then reduced followed by another roll-over to right the aircraft and a final pull-up (out). The resulting rectilinear accelerations are shown in figure 5 and the angular accelerations in figure 6.

Inverse Dynamics Implementation

A form of model-based control referred to as in-

verse dynamics[5] has been implemented as a controller alternative to PID compensation in the centrifuge model to demonstrate its capability to improve response. Inverse dynamics is particularly suited to this application since it uses an inverse model of the dynamics of the centrifuge to 'anticipate' and 'compensate' for nonlinear coupling effects between the planetary drive command and the roll and pitch pointing commands of the cab.

The basic equations of motion (EOM) of the GFET have been developed symbolically under the assumption that all links are rigid. The model includes bearing friction and aerodynamic drag. The EOM are given in vector (matrix) form by:

$$\tau = M(q)\ddot{q} + C(\dot{q},q)\dot{q} + G(q) + \begin{bmatrix} \tau_d \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \tau_{fx} \\ \tau_{fy} \\ \tau_{fz} \end{bmatrix} \quad (8)$$

where q is the vector of joint angles $(q_1, q_2, q_3)^T$, τ is the vector of torques applied to the three joints by motors/gear reducer sets, $M(q)$ is the 3x3 inertia matrix, $C(\dot{q},q)$ is the 3x3 matrix of non-linear Coriolis and centrifugal torque coefficients, $G(q)$ is the torque vector due to gravity, τ_d is torque applied to the planetary axis by aerodynamic drag and τ_{fi} is the ith component of torque applied to the joints by bearing friction.

A model of the plant in the math model of the DFS GFET (the plant module in figures 2 and 4) was implemented by double integrating the joint accelerations $\ddot{q}_i$ obtained from equation 8:

$$\ddot{q} = M^{-1}(\tau - C - G - \tau_d - \tau_f) \quad (9)$$

given values of commanded torque and initial conditions for state variable q. This is referred to as a forward dynamic solution. Inverse dynamics is implemented in the controller of the math model by using equation 8 directly to compute a torque command given estimates of the joint variables q_i and their derivatives.

The inverse control law is defined to be $\tau \doteq \tilde{M}v + \tilde{C}\dot{q} + \tilde{G} + \tilde{\tau}_d + \tilde{\tau}_f$ with $v \doteq \ddot{q}^{des} + K_1(\dot{q}^{des} - \dot{q}) + K_0(q^{des} - q)$ where q_i^{des} is the desired value for state variable q_i. When this definition is substituted in equation 8 and it is assumed that the 'tilde' terms are precisely equal to the corresponding 'non-tilde' terms of equation 8 (i.e. when the model to be used in the controller is precise), one obtains:

$$v = \ddot{q} \quad (10)$$

and

$$\ddot{e} + K_1\dot{e} + K_0 e = 0 \quad (11)$$

where $e = (q^{des} - q)$. The implication of equation 10 is that the components of the control variable components v_i that have been introduced are decoupled and each is linearly related to its corresponding component of acceleration. Equation 11 indicates that the error e in the controller can be driven to zero at a speed determined by the gain settings of diagonal matrices K_1 and K_0. It is assumed that these implications also apply if the model used in the controller is 'nearly precise'.

To implement this control law, it is necessary to have q^{des}, $\dot{q}^{des}$, and $\ddot{q}^{des}$. The first, q^{des}, is calculated using equation 4. The first and second derivatives of q are obtained by differentiating equation 4 with respect to time. The equations that result have functional dependencies as follows:

$$\dot{q}_2 = \dot{q}_2(\dot{q}_1, \ddot{q}_1, q_2, y_v^{(3)}) \quad (12)$$

$$\ddot{q}_2 = \ddot{q}_2(\dot{q}_1, \ddot{q}_1, q_1^{(3)}, q_2, \dot{q}_2, y_v^{(4)}) \quad (13)$$

$$\dot{q}_3 = \dot{q}_3(\dot{q}_1, \ddot{q}_1, q_1^{(3)}, q_2, \dot{q}_2, q_3, x_v^{(3)}) \quad (14)$$

$$\ddot{q}_3 = \ddot{q}_3(\dot{q}_1, \ddot{q}_1, q_1^{(3)}, q_1^{(4)}, q_2, \dot{q}_2, \ddot{q}_2, q_3, \dot{q}_3, x_v^{(4)}) \quad (15)$$

where the superscripts (3) and (4) indicate the third and fourth time derivatives, respectively. In implementation, all derivatives above the third in x and y were set to zero. Third derivatives of x_v and y_v that are required for the above equations are obtained by differentiating the output of the aerodynamic model. The first derivative, $\dot{q}_1^{des}$ can be derived from equations 1-3 for steady state operation as:

$$\dot{q}_1^{des} = \sqrt[4]{(\ddot{x}_v^2 + \ddot{y}_v^2 + \ddot{z}_v^2 - g^2)/L_1^2} \quad (16)$$

The second, third and fourth derivatives of q_1 are obtained by differentiating equation 16. A block diagram representation of the model with an inverse dynamics controller is shown in figure 7 where equation 8 is evaluated in the block labeled 'Inverse Dynamics' and equations 12-15 are evaluated in the blocks labeled 'Determine $q_i\dot{q}_i\ddot{q}_i$'.

Figures 8, 9, 10 and 11 compare the model response of the PID and a precise inverse dynamic (ID) controller applied to the sample mission. Both the PID and ID controllers track the desired G_z command relatively well, as indicated in figure 8, with the exception of commands that are less than the g-level associated with centrifuge idle speed, that for this model of a Generation I GFET machine is 1.4g.[4] There is however, significant difference in delay for the two controllers. Figure 9 is a magnification of

[4] Generation II GFET machines are able to invert the occupant to simulate both positive and negative g-levels created in aircraft 'push-pull' maneuvers.

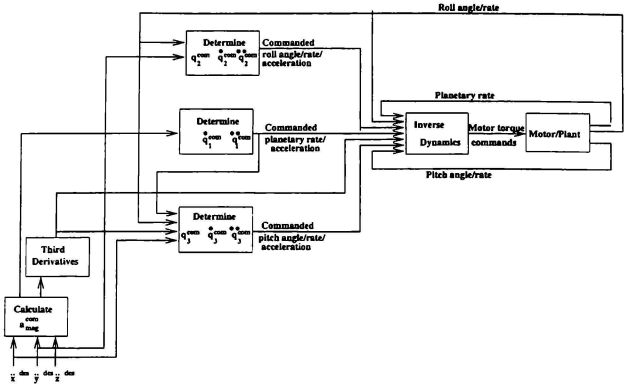


Figure 7: Block diagram of the G-pointing control architecture that uses inverse dynamic control.

the G_z response in the vicinity of 3 secs where the delay is seen to be approximately 300 ms for the PID controller and approximately 50 ms for the ID controller. Predicted PID delay is close to the measured delay of the GFET machine stated in figure 1a. Magnitude error is greater for the ID controller but that is of less consequence in assessing response fidelity. There are two reasons to expect response improvement when ID control is used:

- Velocity error and desired acceleration supplement the error in angle as terms which produce control action.
- nonlinear dynamic coupling of state variables is included in the control action.

Some degradation of performance can be expected when the dynamic model used in the ID controller does not precisely match that of the plant or there is noise in the system. The ID controller was run with $\hat{C} = \hat{G} = \hat{\tau}_d = \hat{\tau}_f = 0$, i.e. only centrifuge inertia was included in the inverse dynamic model. The results are presented as curve 'Inertia only' in figure 12 and are seen to be nearly as effective as when the full model is used. Likewise, the curve labeled 'Imprecise inertia only' in figure 12 was generated with the principle moments of inertia of the planetary arm in error by 5% and cannot be discerned from the 'Inertia only' curve. This data indicates that there should

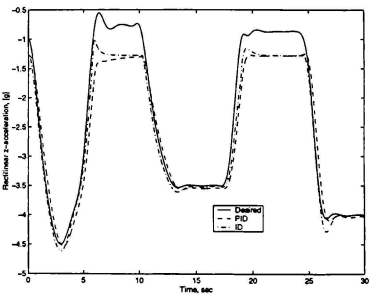


Figure 8: Comparison of G_z response for the PID and ideal inverse dynamic controllers applied to the sample mission.

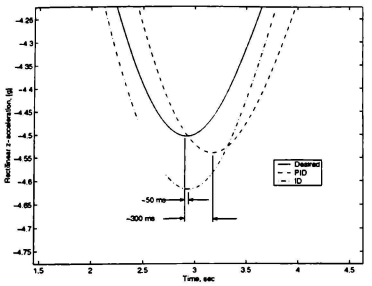


Figure 9: Comparison of time lag in G_z response for the PID and ideal inverse dynamic controllers applied to the sample mission.

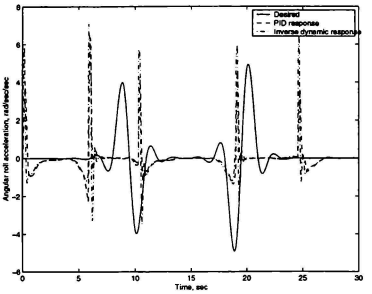


Figure 10: Comparison of roll acceleration artifacts for the PID and ideal inverse dynamic controllers applied to the sample mission.

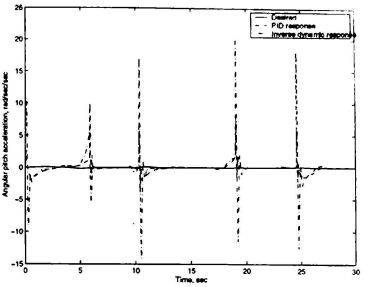


Figure 11: Comparison of pitch acceleration artifacts for the PID and ideal inverse dynamic controllers applied to the sample mission.

be little degradation in controller performance for the expected degree of accuracy with which the inertial properties of the centrifuge can be estimated. The effect of noise is not discussed in this paper.

The most significant artifacts in figures 10 and 11 are seen to be the large spikes that occur at approximately 6, 10.5, 19 and 25 seconds. These are the result of transitioning into or out of level flight.[5] Of course, the desired 180 degree roll maneuvers at approximately 9 and 19 seconds (see figure 6) are missed completely by both the PID and ID controllers. While difficult to discern from the figures, the peak magnitude of the PID spikes are approximately $\frac{1}{2}$ of the ID magnitudes as would be expected since ID exhibits tighter control of pointing. The peaks become larger, but their duration becomes shorter as the g-pointing errors are reduced. This begs the question of whether such accelerations, not unlike spikes present at roll onset of a high-performance aircraft, are disturbing to the rider. Tests have been conducted, as described below, in order to answer this question.

Human Sensitivity to Centrifuge Artifacts

The objective of the tests described herein was to develop a subjective assessment of human perceptual

[5]Large angular cab excursions are required because the tangential component of centrifuge acceleration becomes more influential in determining direction of the resultant centrifuge acceleration vector.

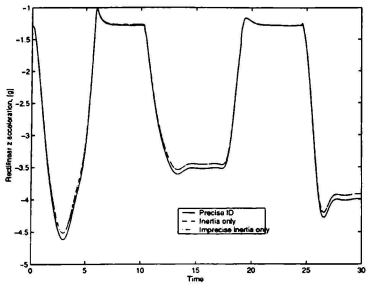


Figure 12: Comparison of G_z response for abbreviated and imprecise forms of inverse dynamic control applied to the sample mission.

sensitivity to artifacts of angular and linear acceleration at various G loadings. The angular artifacts investigated are similar to the spikes described in the paragraph above for tight g-pointing control. The linear artifacts are G_x and G_y pulses that would result if the precision of the g-pointing controller was degraded. The tests were conducted with no task distractions in order to make the subjects more sensitive to the artifacts.

Overall Equipment Set-Up- The Dynamic Environment Simulator (DES), a man-rated centrifuge at Wright Patterson AFB, was set up in two different configurations; seat facing forward in a tangential direction for Gy (lateral) exposures and seat facing inboard radial for G_x (fore and aft) exposures. The visual field consisted of a projected text on a white screen in a dark cab and operation of the flight stick. The seat had a 30^0 seat back angle. The DES arm speed and cab position was under open loop computer control with hyper-gravity experienced in two axes simultaneously. G_z onset rate was the maximum obtainable with arm torque (approximately 0.75 Gz/sec).

Subjects- The test subjects were volunteer members of the DES Sustained Acceleration Research Panel, having passed all required medical screening and completed indoctrination training. They were trained in the verbal responses required for measurement.

Experimental Procedures- Subjects were exposed to a series of 0.75 G_z/s onset ramps to plateaus at 1.4, 2, 4, 6, and 8 G_z as well as a control condition at 1 G_z. The plateau lasted 12 seconds. During the plateau, they also experienced a Gy or G_x pulse. The pulse was sustained for 4, 6 and 8 G_z plateaus or momentary for 1, 1.4, and 2 G_z. The magnitude of the pulse was varied according to Table 2 and the onset of the pulse was at a set alpha rate of 1, 2, 4, 7, or 10 radians per second squared, also found in Table 2 (note each separate test profile is designated by a letter).

Experiment	Profiles Used (Fixed Factor)
1	v,w,x,c,f,ii,b,e,h,a,d,g ($G_x = 0.5$)
2	ii,l,h,k,g,j ($G_z = 2$)
3	m,n,o,p,q,r,s,t,u ($\alpha = 4.0$)
4	y,z,m,o ($G_z = 4.0$)

Table 3: Profiles used in each experiment (see also table 2).

There were 7 subjects, each experiencing 3 repetitions of the tests described above. After returning to baseline, subjects verbally responded with a numerical indication of the intensity of the perceived artifact. The artifact response rating (ARR) scale was:

- 0 = did not feel at all,
- 2 = noticed it
- 4 = felt but not disruptive
- 6 = felt and caused some distress
- 8 = felt and caused significant discomfort
- 10 = totally unacceptable

G_x Results- The 26 profiles were used in 4 separate experiments in which one of the factors (α, G_z, or G_x) was fixed at one level so that an analysis could be performed for the other 2 factors. The dependent variable was the mean rating across the 3 repetitions for each subject and profile. These four experiments, referred to as Experiments 1 to 4, are defined as shown in the Table 3.

Figure 13-16 show the interactions of the three conditions of G_z, G_x, and α. Figure 13 shows that the ARR was not affected by α, as long as G_x was low. Figure 14 shows that ARR was unaffected by α magnitude but increased with increased G_x when G_z was low. In other words, it was the G_x that bothered subjects, not the rate of the pitching motions. Figure 15 shows that ARR is a function of both rectilinear components, but in opposite directions; showing more sensitivity when in high G_x and less sensitivity when in high G_z. Figure 16 shows that even at

	Low G									High G								
G_z	1 (control)			1.4			2			4			6			8		
G_x or G_y	0.5	1	1.5	0.5	1	1.5	0.5	1	1.5	0.5	1	1.5	0.5	1	1.5	0.5	1	1.5
$\alpha(r/s^2)$																		
1											y		z					
2	v			w			x											
4	c			f			ii	l		m	n	o	p	q	r	s	t	u
7	b			e			h	k										
10	a			d			g	j										

Table 2: Letter designation of the 26 combinations of conditions tested.

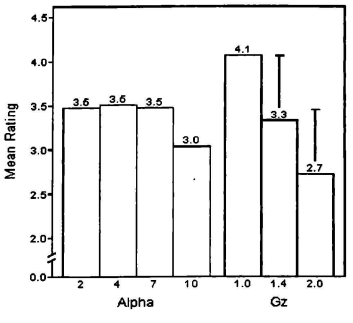


Figure 13: Experiment 1 mean artifact response ratings. Whiskers are minimum significant difference from the Bonferroni paired comparison procedure ($G_x = 0.5$).

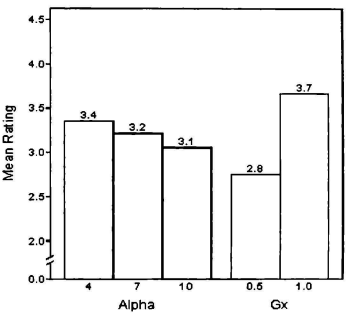


Figure 14: Experiment 2 mean artifact response ratings ($G_z = 2.$).

moderate G_z level, ARR is not sensitive to α, but still shows response to the G_x component. These results suggest that high angular artifacts need not be a serious concern in DFS design. All rates from 1 to 10 radians/second/second were rated nearly the same subjectively, regardless of the G-level attained. Subjects showed considerably more sensitivity to increasing G_x bias than to increasing angular acceleration spikes and high G_z somewhat masks both effects.

Repeated measures analyses of variance were performed using subject interactions as error terms. Post-hoc paired comparisons used the Bonferroni paired comparison procedure with a family-wise er-

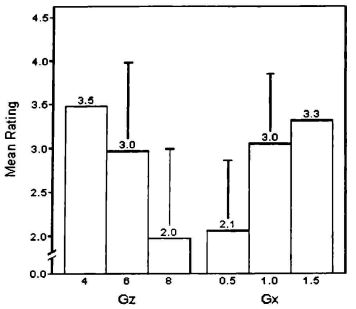


Figure 15: Experiment 3 mean artifact response ratings. Whiskers are minimum significant difference from the Bonferroni paired comparison procedure ($\alpha = 4.$).

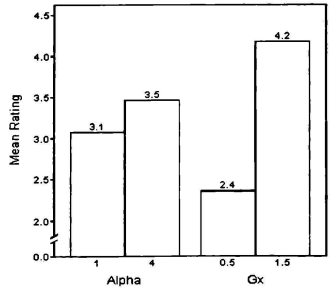


Figure 16: Experiment 4 mean artifact response ratings($G_z = 4.$).

Increase in factor	Effect of Sensitivity to the artifact
α	None
G_z	Decrease
G_x	Increase

Table 4: General findings of the G_x analysis.

Increase in factor	Effect of Sensitivity to the artifact
α	None
G_z	Decrease
G_y	Increase

Table 5: General findings of the G_y analysis.

ror level of 0.05. A summary of the results is contained in Table 4. Though not statistically significant, there was a trend for faster alpha to be preferred at lower G_z levels.

G_y Results- Method of experimental design and analysis were identical to the G_x portion of the experiment. Table 5 shows the results for the lateral artifacts. The trend for faster alpha to be preferred at lower G_z levels was not observed in the G_y artifacts.

Test Summary- It was anticipated that very high alphas, such as 10 radians per second squared would be deeply disturbing and possibly biodynamically dangerous. However, this was not at all the finding. The results indicate the effect of these artifacts to be no more than a mild disturbance over the expected range of g-loading and artifact magnitudes and, hence, seem to substantiate anecdotal information from riders of the GFET Generation I centrifuge in which no artifact disturbances have been noted. The tests also suggest that for the range of accelerations tested, precision in G_x and G_y, as G-magnitude increases, is more important than is precision in angular acceleration. Test repetitions with subjects showed no change in responses, i.e. no learning or conditioning effects were noticed. Specific results of the tests are:

- G_y at high G_z is the most painful artifact. Controls to avoid it should be a top priority.
- High angular accelerations (alphas) were slightly preferred to low alphas, but all alphas were comfortable, an unexpected result.
- Subjects preferred lower G_x and G_y to higher ones significantly.

- Quicker transitions were preferred, especially at low G_z. At high G_x, the G_z seemed to mask this effect.
- Increased discomfort was associated with an increase in the magnitude of the artifact in the G_x direction, and more so in the G_y direction.

There appears to be no biodynamic reason to preclude the application of high fidelity, rapid acting, high torque gimbals to enable close matching of the rectilinear requirements.

Conclusion

The following conclusions can be drawn concerning the development of a sustainable-G dynamic flight simulator:

- Data suggests a centrifuge of the GFET class with an inverse dynamic controller may be able to respond with less than 300 ms of delay as recommended by the FAA for simulator category A/B and some GFET machines may be able to respond with less than 150 ms of delay as recommended by the FAA for simulator category C/D.
- Human response test data suggests a centrifuge occupant will not be disrupted by the angular acceleration artifacts associated with rapid response.
- Data shows that occupant sensitivity to angular acceleration decreases slightly at higher g exposure whereas sensitivity to G_x or G_y increases-this suggests precise g-pointing may be a desirable control strategy.

The results will be implemented on high performance centrifuges built by ETC to validate the predictions.

Recommendations for future work:

- Human response tests to establish suitable lag limits at elevated g-levels.
- Human response tests to establish suitable drive algorithms for α cues that account for cross-coupling impact on perceived response.
- Hardware validation of inverse dynamics ability to improve centrifuge response.

Credits

This work has been accomplished collaboratively under Cooperative Research and Development Agreements (CRDA) 96-AFIT-02 between Environmental Tectonics Corporation (ETC) and the Air Force Institute of Technology and CRDA 99-141-HE-01 between ETC and the Human Effectiveness Directorate of the Air Force Research Laboratory.

References

[1] -, "Airplane Simulator Qualification," Advisory Circular No. 120-40B, Federal Aviation Administration, June 6, 1993.

[2] Brown, Yorke J., F.M. Cardullo, G.R. McMillan, G.E. Reccio and J.B. Sinacori, "New Approaches to Motion Cuing in Flight Simulation," Cardullo, Brown and Associates, Air Force Research Laboratory Technical Rpt. AL-TR-1991-0139, Sept 1991.

[3] Crosbie, Richard J., and Dennis A. Kiefer, "Controlling the Human Centrifuge as a Force and Motion Platform for the Dynamic Flight Simulator Technologies," Proceedings of the AIAA Flight Simulation Conference, St Louis, MO, AIAA Paper 85-1742, July 1985.

[4] Poppel, J.A., B.J Barton, D.j. Pancratz, M.H. Rangel, R.D. Banks, and J.B. Bomar, Jr, "Simulation of Thrust-Vectored Aircraft Maneuvers in a Human Centrifuge: Model Validation and Design for the Dynamic Environment Simulator," Biodynamics Research Corporation, Air Force Research Laboratory Technical Rpt AFRL-HE-WP-TR-1998-0138, Sept 1998.

[5] Spong and Vidyasagar, *Robot Dynamics and Control*, Wiley, 1989.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37218

AIAA-2000-4076

SIMULATED LANDINGS IN TURBULENCE – MOTION, PREDICTIVE MODELLING, AND PSYCHOMETRIC ASPECTS*

E. Svensson, Ph.D., & M. Angelborg-Thanderz, Ph.D
National Defence Research Establishment (FOA),
Linköping, Sweden †

Abstract.

To maintain control in approach/landings under turbulence, the pilot requires prompt attention. The lead provided by non-visual cues is then important. One objective of two studies was to analyse whether a moving base dome simulator could produce more realistic cues than a fixed base simulator. In the first study half of 60 landings were performed with the motion system disengaged. The pilots rated risk, difficulty, workload, performance, handling qualities, and induced oscillations. Stick activity and difficulty were higher and performance lower under the motion condition. In a second study the importance of motion in turbulent landings was verified and analyses of the pilots' control responses showed that there were inter-individual differences. Model analyses showed that turbulence affects workload, and that workload, in its turn, influences performance. We found that handling qualities and induced oscillations could be predicted from the other variables. The variables turbulence and motion of the aircraft explains 65 percent of the variance in handling qualities ratings and 36 percent of the variance in ratings of pilot induced oscillations. Accordingly, handling qualities and pilot induced oscillations can be estimated and predicted in situations 'without man in the loop'. The analyses present psychological aspects related to and underlying aircraft handling qualities and pilot induced oscillations.

General background

Flying is characterised by often extreme motion, and even the simplest simulator presents motion cues to the pilot. The pilot exercises control over his aircraft by sensing, processing, and interpreting information provided by the aircraft, its environment, and by his own control inputs.

Motion can aid the pilot in controlling and manoeuvring the aircraft by confirming that the response of the aircraft matches the pilot's internal model of the expected response (i.e., can give him adequate sensory feed-back), and it can help him to recognise failures of the system. Motions, such as those resulting from turbulence and vibration, also make the pilot's task more difficult and increase his mental workload, perceived risk, and psychological stress. Senses as vestibular, proprioceptive (in muscles, viscera, and joints), and tactile are also stimulated, primarily by motions of the cockpit. Visual cues of motion are probably the most important 'low frequency cues', while the cues from the vestibular, proprioceptive, and tactile senses are the most important 'high frequency cues'. The non-visual receptors respond more quickly than the visual system, since they are sensitive to motion onset rather than to sequential events. In conventional aircraft, manoeuvre cues are of comparatively low frequency, and these cues can generally be provided visually.[1, 2, 3, 4, 5, 6, 7, 8, 9] Repperberger reports that the tactile system reacts about 2.5 times faster than the visual system.[10] Hosman reports similar effects of peripheral vision and the vestibular system.[11] The responses from the vestibular system, the visual system, and effects of their combination have been demonstrated.[12]

How the information from the different sensors is integrated is not known, but the information processes are dependent on the pilot's mental model of the aircraft, the environment, and on his experience and training. Thus, the pilot's experience and training, the frequency response of his various sensors, and his simultaneous reception of signals will affect his perception of motion. His thresholds of linear and angular motion will increase due to mental workload and superimposed vibration.[13]

There have been a great improvement of motion platform systems in recent years, and techniques to measure and analyse pilot performance and pilot mental workload have been improved as well.[14, 15] Because of these developments, new research may provide indications that non-visual motion cues are effective for specific flight tasks, e.g., landings in turbulence. The importance of using good aircraft motion cueing, when evaluating critical handling qualities, has been explain-

* This paper presents the results of two studies performed at NLR by Saab AB in co-operation with Defence Research Establishment. Mr. E. Kullberg, Saab, and Mr. J. Ersson, FOA, have made contributions to conclusions and analyses, respectively.

† Copyright © 2000 by Defence Research Establishment, Sweden. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

ed by White, Hall & Tomlinson.[16] The elimination of critical handling problems before a vehicle flies would be highly cost effective. From AGARD we can conclude that instability of the pilot/airframe configuration or pilot induced oscillations is still a critical handling quality factor of modern aircraft.[17] Deviations from real flight with respect to realism can affect the pilot's behaviour. "If pilots perceive vehicle motions accurately, their control responses will be the same as if they were flying the real aircraft".[18]

The main objective of the first study was to test whether a moving base dome simulator would produce more realistic and useful cues to the pilot (i.e., give him better sensory feedback) than a fixed base simulator in simulations of turbulent landings with a high performance aircraft.

The objectives of a second study were (a) to validate the results of study one, (b) to analyse interindividual differences in pilots' control behaviour during simulated landings, (c) to analyse effects of landing sequence on fatigue, control behaviour, and performance, (d) to test a causal model of pilot performance during turbulent landings (e) to predict handling qualities and pilot induced oscillation ratings, and (f) to develop practicable measures of workload and performance.

Study one

Method

A repeated measurement design was used, and two experienced test pilots performed 30 simulated offset landings each under five different levels of turbulence. Half of the landings were performed with the motion system disengaged. The study was performed with a modified F16 aircraft model. A centre mounted JAS39 control stick was used.

Directly after each landing the pilots rated six different aspects of the task: 'risk of crash during approach/landing' (RISK), 'difficulty in manoeuvring the aircraft' (DIFFIC), 'pilot mental workload' (PMWL), 'pilot performance' (PERF), 'aircraft handling qualities' (CHR), and 'pilot induced oscillations' (PIOR). The first four questions were specified in detail and answered on Likert scales with anchored endpoints. These types of items have been used in a series of studies of military pilots. [15, 19, 20, 21, 22, 23, 24] 'Aircraft handling qualities' were estimated by means of the Cooper-Harper Scale and 'pilot induced oscillations' by means of the PIO Tendency Rating Scale.[16, 18 25] The results are based on these rated variables, the independent variable 'turbulence' (TURB), and the pilots' 'stick activity' (STICK). Primarily, correlational statistics (simple and multiple regression) and multidimensional scaling (MDS) were used.

Results

Differences between means. The pilots differed consistently on all variables except 'performance' (PERF). Pilot B had significantly higher 'stick activity', and he rated 'risk for an accident', 'difficulty', 'mental workload', and 'pilot induced oscillations' higher and 'handling qualities' lower than pilot A. The standard deviations were higher for pilot B than for pilot A.

From means and standard deviations for the test conditions 'no motion' (NM) and 'motion' (M1) and significance levels for differences between the conditions for 'stick activity' and the six rated variables, we found that the 'stick activity' and the 'difficulty' were higher, and the 'performance' level lower during 'motion'. The 'handling qualities' tended to be lower, and the 'risk of an accident' tended to be higher. Except for 'pilot induced oscillations' all differences were in the expected direction. Our conclusion is that motion makes the landing task harder, resulting in increased 'stick activity' **and** decreased 'performance'.

Variable intercorrelations. The correlation coefficient is a robust measure, insensitive to differences in means and skewness of distributions, and its power is stronger than that for first order statistics such as the mean. It was found that there were significant differences between the two pilots in 'stick activity' and the six rated variables. But do the correlation structures, i.e., the relationships between the seven variables, differ between the two pilots?

Table 1. Product moment correlations between stick activity and the six rated variables for pilot A (lower triangular matrix) and pilot B (upper triangular matrix), respectively. The diagonal values (in italics) present the concordance between the pilots (r-values > .38 are significant at p-level = .05).

	RISK	DIFF	PMWL	PERF	STICK	CHR	PIOR
RISK	*+0.599*	+0.986	+0.968	-0.777	+0.743	+0.900	+0.722
DIFF	+0.719	*+0.758*	+0.990	-0.694	+0.753	+0.918	+0.733
PMWL	+0.640	+0.853	*+0.787*	-0.694	+0.753	+0.918	+0.773
PERF	-0.161	-0.600	-0.573	*+0.380*	-0.593	-0.692	-0.410
STICK	+0.598	+0.801	+0.795	-0.516	*+0.771*	+0.725	+0.484
CHR	+0.595	+0.840	+0.720	-0.652	+0.816	*+0.584*	+0.779
PIOR	+0.388	+0.181	+0.035	+0.186	+0.106	+0.124	*-0.194*

Table 1 presents the correlation structures (with **and** without motion) for pilot A (the lower triangular matrix) and pilot B (the upper triangular matrix). We have used the correlation *between* the 21 coefficients for pilot A and for pilot B as a similarity measure. The two correlation structures of pilot A and B were quite

similar ($r = .86$; $p < .001$), even if the correlations were significantly higher for pilot B ($t = 4.72$; $p < .001$). The higher variable variance of pilot B may explain this difference. The diagonal values (in italics) of the table present the consensus between the two pilots for each variable under identical conditions. As shown by the diagonal values, the concordance is significant ($p < .05$) for all variables except for 'pilot induced oscillations' (PIOR). Thus, in spite of significant pilot differences in means and variances, the correlation structures for pilots A and B are in accordance as well as their 'stick activities' and ratings of the specific items (the diagonal values).

The individual differences in means and variances have been controlled for by means of Z-transformations ($M = 0.00$, $Sd = 1.00$) of the variables of pilot A and pilot B, respectively. The correlation structure based on these transformed data has been compared with the structure based on the raw data, both pilots pooled. The correspondence between the two structures is impressive ($r = .98$; $p < .001$); the common variance is 96 percent. This finding illustrates the insensitivity of the correlation coefficient to the differences in means between the pilots. Accordingly, the significant concordance between the two correlation structures warrants a combination of the data from both pilots. Thus, the structures, to be presented, will be based on data from pilot A **and** pilot B.

Table 2. Product moment correlations between stick activity, the six rated variables and turbulence level (r-values > .28 are significant at p-level = .05). The covariations are based on data from both pilots and both conditions ['motion' (M1) and 'no motion' (NM)].

	RISK	DIFFIC	PMWL	PERF	STICK	CHR	PIOR	TURB
RISK	1.000							
DIFFIC	+0.838	1.000						
PMWL	+0.875	+0.951	1.000					
PERF	-0.557	-0.598	-0.585	1.000				
STICK	+0.708	+0.803	+0.802	-0.462	1.000			
CHR	+0.870	+0.803	+0.825	-0.514	+0.763	1.000		
PIOR	+0.789	+0.525	+0.587	-0.201	+0.459	+0.754	1.000	
TURB	+0.559	+0.719	+0.713	-0.368	+0.696	+0.594	+0.235	1.000

Table 2 presents the correlation structure based on data from both pilots and both conditions ['no motion' (NM) **and** 'motion' (M1)]. As can be seen, all relationships exept two (the correlations between PIOR and PERF, and between PIOR and TURB) are significant. The mean correlation is high, and 68 percent of the total variance can be explained by one common factor.

The relationships have been analysed by means of distance weighted least squares regression. From these analyses we found that some relationships were curve-linear. Figure 1 illustrates the relationship between 'risk of crash' during landing (RISK) and 'aircraft handling qualities' (CHR).

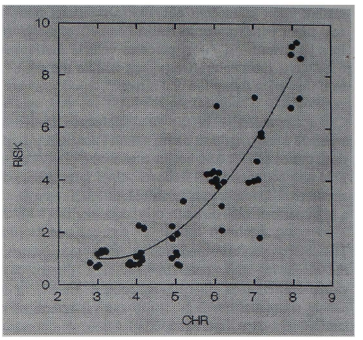


Figure 1. The perceived risk of crash (RISK) as a function of aircraft handling qualities (CHR). The relationship is based on the complete data base (pilot A and pilot B with and without motion). The curve is fitted by means of distance weighted least squares regression.

The exponential function $RISK=EXP(-1.515 + 0.453 * CHR)$ describes the curve.

Table 3 (next page) presents the correlation structures for the condition 'no motion' (NM) and 'motion' (M1), respectively. There is a close correspondence between the structures besides the relations between 'performance' (PERF) and the other variables. No differences, besides the correlations with 'performance', are significant.

Table 3. Product moment correlations between stick activity, the six rated variables, and turbulence level under 'no motion' (NM) (lower triangular matrix) and 'motion' (M1) (upper triangular matrix) (r-values > .40 are significant at p-level = .05). The co-variations are based on data from both pilots.

	RISK	DIFFIC	PMWL	PERF	STICK	CHR	PIOR	TURB
RISK		+0.824	+0.881	-0.706	+0.701	+0.868	+0.844	+0.602
DIFFIC	+0.857		+0.960	-0.714	+0.803	+0.810	+0.575	+0.779
PMWL	+0.841	+0.926		-0.711	+0.853	+0.857	+0.652	+0.770
PERF	+0.135	-0.044	-0.079		-0.525	-0.705	-0.508	-0.471
STICK	+0.787	+0.825	+0.805	+0.056		+0.803	+0.562	+0.776
CHR	+0.913	+0.772	+0.764	+0.073	+0.714		+0.763	+0.636
PIOR	+0.868	+0.559	+0.577	+0.282	+0.511	+0.822		+0.321
TURB	+0.514	+0.664	+0.645	-0.174	+0.759	+0.537	+0.148	

Multidimensional scaling (MDS). Figure 2 a presents a two-dimensional MDS-solution for the data from the 'no motion' (NM) condition.[26] As expected from the correlation structure, the 'performance' variable (PERF) is an outlier, unrelated to the other variables. The other variables form an approximate simplex structure which means that they can be ordered in a sequence: **turbulence → stick activity → difficulty → pilot mental workload → risk for an accident → aircraft handling qualities → pilot induced oscillations.**

Figure 2 b illustrates a two-dimensional MDS-solution for the data from the 'motion' (M1) condition. In this solution we found the sequence: **turbulence → stick activity → difficulty → pilot mental workload → aircraft handling qualities → performance → risk for an accident → pilot induced oscillations.** This sequence forms an embryo to a model of the internal causal relationships between important aspects of the approach/landing situation

Conclusions from study one

When comparing the experimental conditions 'no motion' and 'motion', we found that the 'stick activity' and the 'difficulty level' were higher, and the 'performance level' lower during motion. The 'handling qualities' tended to be lower, and the perceived 'risk of an accident' tended to be higher during the same condition.

Thus, in the flight task analysed in study one, the pilot's stick inputs were reduced, when the motion system was disengaged.[9] Accordingly, from the reduced stick inputs under these conditions, one runs the risk of overestimating the handling qualities of the real aircraft.

All correlations between 'performance' and the other variables were significant under the 'motion' condition, but **none** were significant under the 'no motion' condition. These findings are of special importance with respect to the questions addressed in the study. The pilot's acquisition of skill is based on his performance feedback, i.e., his ability to correctly perceive his performance. Lack of relevant cues diminishes his possibility to get performance feedback. Furthermore, this lack of relevant non-visual cues makes a positive transfer of training from simulations to real landings less likely.

We have used multidimensional scaling (MDS) fitting our variables in a space in such a way that the distances between them correspond to their inter-correlations. In the scaling of the data from the 'no motion' condition, we found, as expected from the correlation structure, that the 'performance' variable was an outlier unrelated to the other variables. Furthermore, we found that the other variables formed an approximate simplex structure. During the 'motion' condition the variables were ordered in a sequence, forming an embryo to a model of the internal relationships between important aspects of the approach/landing situation. It is of special interest that the independent variable 'turbulence'

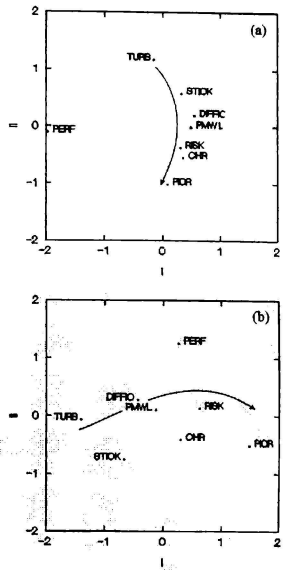


Figure 2. Two-dimensional MDS solutions for data from the 'no motion' (NM) (a) condition and the 'motion' condition (M1) (b). According to Guttman-Lingoe's coefficient of alienation, 99% of the variance in the data is explained by the solutions.

forms the starting point of the sequence and that it is followed by the objective measure of 'stick activity'. Furthermore, there is no **direct** relationship between turbulence and 'pilot induced oscillations' but a significant **indirect** one, mediated by 'stick activity' and the other aspects.

Study two

Method

Four experienced test pilots performed 48 simulated landings each under five different levels of turbulence. Because of economical and practical reasons all landings were performed with the motion system engaged. Accordingly, a direct comparison between the 'motion' and 'no motion' conditions could not be performed. Instead the correlation matrix from the 'motion' condition of this study (M2) was compared with the corresponding matrices ['no motion' (NM) respectively 'motion' (M1)] in the first study. Directly after each landing the pilots rated the same aspects as in the first study, as well as their 'fatigue' and 'psychological stress'. Furthermore, a combination of aircraft motions (in pitch, yaw, and roll) at touch down was added as an objective measure of performance. The study was performed with an F16 aircraft and a JAS39 model of flight control. Correlational statistics and Linear causal modelling was performed by means of LISREL.[27, 28] This procedure makes possible a statistical test of the validity of causal flow models.

Results

Variable intercorrelations. Table 4 presents the correlation structure based on the four pilots' 192 landings .

Table 4. Product moment correlations between stick activity, the six rated variables and turbulence level (r-values > .27 are significant at p-level < .001). The correlations are based on 192 landings (with motion system engaged) of four pilots.

	RISK	DIFFIC	PMWL	PERF	STICK	CHR	PIOR	TURB
RISK	1.000							
DIFFIC	+0.893	1.000						
PMWL	+0.753	+0.858	1.000					
PERF	-0.567	-0.634	-0.533	1.000				
STICK	+0.693	+0.788	+0.734	-0.499	1.000			
CHR	+0.860	+0.867	+0.719	-0.689	+0.721	1.000		
PIOR	+0.687	+0.644	+0.516	-0.576	+0.594	+0.725	1.000	
TURB	+0.736	+0.807	+0.708	-0.501	+0.757	+0.787	+0.582	1.000

As can be seen, all relations are significant, and the mean correlation is high.

When comparing the matrix of the 'motion' condition (M1) of study one (table 3) with the corresponding matrix (M2) of study two (table 4), we found a close correspondence. The mean difference between the correlations is insignificant (mean = .005, standard deviation = .106). The correlation between the structures is .98 ($p < .001$). Thus, there was a close concordance between the 'motion' conditions of the two studies. Furthermore, the matrix of study two (M2) showed the same differences to the 'no motion' matrix (NM) as the 'motion' matrix (M1) of study one. Performance (PERF) was significantly related to the other variables under the 'motion' conditions (M1 and M2) but unrelated to the other variables under the 'no motion' condition (NM). Figure 3 illustrates these differences between the 'no motion' condition (NM), and the 'motion' conditions (M1 and M2).

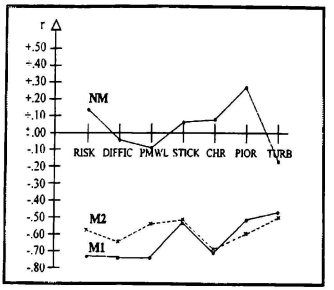


Figure 3. Correlations between performance (PERF) and the other variables for the three conditions 'no motion' (NM), 'motion' study one (M1), and 'motion' study two (M2). All correlations are significant for the conditions M1 and M2, and none is significant for the condition NM.

Interindividual differences in 'stick activity'. The pilots' 'stick activity' has been analysed by means of repeated measurement analyses of variance. In a first analysis, comprising the whole turbulence spectrum, we found a significant group-effect ($F_{(3)}=19.22$, $p < .001$). This means that the pilots differ with respect to 'stick activity'.

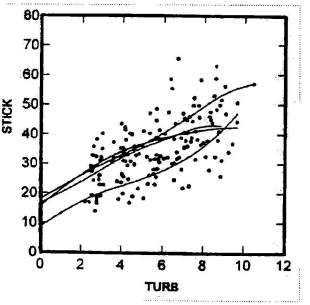


Figure 4. Stick activity (STICK) in degrees/second as a function of turbulence (TURB) in metres/second. Each curve represents a pilot.

Figure 4 illustrates the inter-individual differences as well as the significant increases in 'stick activity' (STICK) as a function of 'turbulence' (TURB) ($F_{(5)}$= 62.99, $p < .001$). Each curve represents a pilot. As can be seen from the figure, one pilot deviates from the three other. The group effect of the ANOVA disappears if we exclude this pilot. No significant interactions were found. This means that the pilots increase their 'stick activity' in the same way over the turbulence spectrum. However, when analysing the upper turbulence spectrum (TURB = 6-12), we found a significant interaction ($F_{(3)}$=2.92, p=0.045). This means that there are differences between the pilots' increase in gain. Two of the pilots present a linear increase and two a decelerated increase in gain.

Trends. The effects of the 'landing sequence' on 'fatigue', 'stick activity', and 'performance' were analysed by means of partial correlation and multiple regression analyses. A partial correlation represents the linear relationship between two (or more) variables, when the effects of 'third variables' are nullified or held constant.

The correlation between 'fatigue' and 'landing sequence' is .40 ($p < .001$). This means that 'fatigue' increases as a function of 'landing sequence'. But, there is a strong relation between 'turbulence' and 'landing sequence'. This implies that the correlation between 'fatigue' and 'landing sequence' is caused by the 'turbulence' and not by the 'landing sequence' in itself.

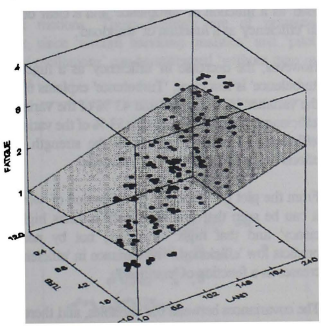


Figure 5. Changes in fatigue (FATIGUE) as a function of landing sequence (LAND) and turbulence (TURB). The dots represent the x, y, and z co-ordinates in a Euclidean space, and the plane represents the optimal 'least squares regression' fit to the dots.

There is a negative relation (-.37, $p < .001$) between perceived 'performance' (PERF) and 'landing sequence' (LAND), which implies that the 'performance' gets lower as a function of 'landing sequence'. However, the partial correlation between 'performance' and 'landing sequence' (with 'turbulence' held constant) is .17, which implies that There is an **increase** in 'performance' as a function of 'landing sequence'. This conclusion is confirmed by a regression analysis showing that there is a positive and significant β-weight of .29 ($p = .020$) for 'landing sequence' (LAND). Thus, there is an increase in 'performance' as a function of 'landing sequence', when the 'turbulence' is held constant. The relations between 'fatigue', 'landing sequence', and 'turbulence' are illustrated in figure 5.

In the same way, we found a strong positive relation between 'stick activity' (STICK) and 'landing sequence' (LAND), which implies that the gain in-creases as a function of the 'landing sequence'. But, the partial correlation between 'stick activity' and 'landing sequence' is -.22 (with 'turbulence' level held constant), which implies that the 'stick activity' **decreases** as a function of 'landing sequence'. This conclusion is confirmed by a regression analysis showing that there is a negative and significant β-weight of -.30 ($p < .005$) for 'landing sequence' (LAND). Thus, the 'stick activity' decreases as a function of 'landing sequence' when the 'turbulence' is held constant.

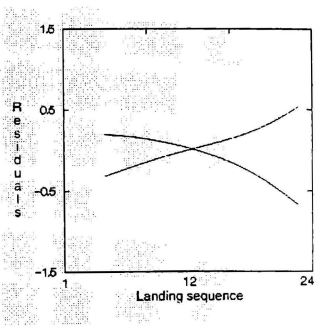


Figure 6. Changes (z-scores) in the residuals for perceived performance (increasing curve) and stick activity (decreasing curve) as a function of landing sequence, when the effects of turbulence are held constant. The curves have been smoothed by means of distant weighted least squares regression.

Accordingly, a genuine increase in 'performance' **and** a genuine decrease in 'stick activity' as a function of 'landing sequence' have been confirmed. The changes in the residuals for 'performance' and 'stick activity' as a function of 'landing sequence' are presented in figure

6. Note that the curves seem to reflect an accelerating learning process.

Causal modelling. In study one, we found that the variables could be ordered in a sequence starting with 'turbulence' and ending with 'pilot induced oscillations'. Our conclusion was that this sequence forms an embryo to a model of the internal causal relationships between important aspects of the landings. The larger database of study two made factor and causal analyses possible and adequate.

In a factors analysis with orthogonal rotation, we found that eight of the manifest variables could be explained by two factors. The two factors explained 72 % of the total variance. Factor one is called **'workload'**, and its manifest markers are 'perceived stress' (STRESS), 'difficulty' (DIFFIC), 'mental work-load' (PMWL), and 'stick activity' (STICK). Factor two is called **'efficiency'**, and its markers are 'handling qualities' (CHR), 'pilot induced oscillations' (PIOR), 'movements of the aircraft at touch down' (MOT. T.D.), and perceived 'performance' (PERF).

In a second step, the causal relationships between the independent variable 'turbulence' and the two factors 'workload' and 'efficiency' were tested statistically by means of Structural Equation Modelling. The final model presented in figure 7 was based on the correlations between nine of the manifest variables. The fit of the model is acceptable [Adjusted Goodness of Fit Index (AGFI) = .81 and the Root Mean Square Residual (RMSR) = .034]. In the model, rectangles indicate manifest variables and ellipses factors. Thin arrows indicate factor loadings and thick arrows effects.

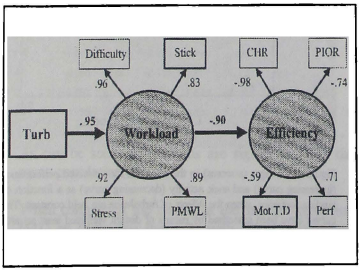


Figure 7. Structural equation model of the relationships between the nine manifest variables. Adjusted Goodness of Fit Index (AGFI) = .81. Rectangles indicate manifest variables and ellipses denote factors. Thick rectangles indicate objective measures. Thin arrows indicate factor loadings and thick arrows causal effects. All effects and factor loadings are significant ($p < .001$).

As can be seen from the model, the independent variable 'turbulence' has a strong effect on the 'workload' factor, and 'workload', in its turn, has a strong effect on the 'efficiency' factor. Thus, increases in 'turbulence' are followed by increases in 'workload', and increases in 'workload' are, in its turn, followed by decreases in 'efficiency'. Thus, 'workload' mediates the effects of 'turbulence' on the 'efficiency'.

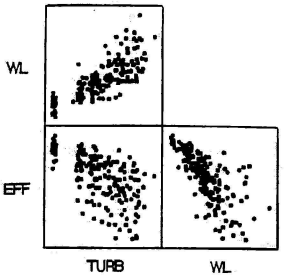


Figure 8. Scatter-plots of the empirical relations between turbulence (TURB), workload (WL), and efficiency (EFF). (n = 192.).

Figure 8 presents the scatter-plots of the empirical values for 'turbulence' (TURB), and the factor points for 'workload' (WL) and 'efficiency' (EFF). As can be seen from the plots, there is a clear increase in 'workload' as a function of 'turbulence' and a clear decrease in 'efficiency' as a function of 'workload'.

However, the decrease in 'efficiency' as a function of 'turbulence' is less clear. 'Turbulence' explains 67 % of the variance in 'workload' and 43 % of the variance in 'efficiency'. 'Workload' explains 53 % of the variance in 'efficiency'. These differences in the strength of the effects are in accordance with the model.

From the plot of 'efficiency' as a function of 'workload' it can be seen that low 'workload' predicts high 'efficiency' and that high 'workload', not by necessity, predicts low 'efficiency'. The variance in 'efficiency' increases as a function of 'workload'.

The covariances between the variables, and thereby the effects in the model, are to a great extent affected by the variance in 'turbulence'. And, as can bee seen from the model, there is a strong negative relation between 'workload' and 'efficiency'. When turbulence is held constant, there is still a negative relation ($r = -.39$, $p < .01$). Thus, there is a genuine negative relation between the factors independent of the level of turbulence. This

negative relation or negative effect of workload on performance has been demonstrated in several studies by the present authors.[15, 20, 24]

When analysing the effects of 'fatigue' we found that 'efficiency' decreases as a function of 'fatigue' ($p < .01$), when 'turbulence' is controlled for. Thus, there is a negative and genuine effect of 'fatigue' on 'efficiency'. No significant and genuine effect of 'fatigue' on 'workload' was found ($p = .08$).

Predictions of 'handling qualities' and 'pilot induced oscillations. Pilot ratings of 'handling qualities' (CHR) and of 'pilot induced oscillations' (PIOR) are critical in the development of flight control systems. Since the sixties the Cooper-Harper scale (CHR) has been used in the evaluation of aircraft handling qualities.[25] What aspects influence the pilots ratings, and are there factors that correlate with pilot ratings of 'handling qualities'? From our regression models we found that CHR and PIOR-ratings can be estimated from other variables. Motions of the aircraft in pitch, yaw, and roll at touch down explains 29 percent of the variance in the pilots 'handling qualities' ratings ($p < .001$). Turbulence level **and** motions of the aircraft in pitch, yaw, and roll explain 65 percent of the variance in the CHR ratings ($p < .001$). In the same way, we found that a weighted estimate of turbulence level and aircraft motions ex-plain 37 percent of the variance in ratings of 'pilot induced oscillations' (PIOR) ($p < .001$). Figure 9 presents the pilot ratings (CHRS) as a function of the weighted estimate (CHREST2) of turbulence level and aircraft motions. These findings are of practical interest, since 'aircraft handling qualities' and 'pilot induced oscillations' can be estimated 'without man in the loop'.

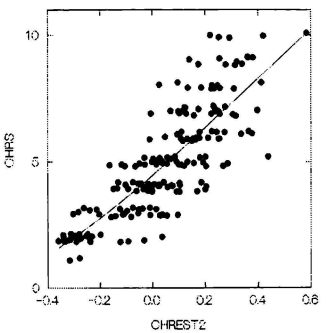


Figure 9. Pilot ratings of 'handling qualities' (CHRS) as a function of a weighted estimate of 'handling qualities' (CHREST2).

Conclusions from study two

When comparing the matrices we found a close correspondence between the 'motion' conditions of the two studies. The relations between 'performance' and the other variables were almost identical under the two 'motion' conditions. These relations are important, because the pilots' acquisition of skill is based on their performance feedback, i.e., their ability to correctly perceive their performance. Our conclusion is that motion produces more realistic and useful cues (i.e., adequate sensory feedback) to the pilot than 'no motion'.

In study one we found significant differences between the two pilots with respect to 'stick activity'. This inter-individual difference in 'stick activity' was verified in the second study. Furthermore, there were differences in the pilots' increase in gain as a function of 'turbulence'. Two of the pilots presented a linear increase in gain, while two presented a decelerated increase, when the 'turbulence' was high.

By means of partial correlation analyses we found a genuine increase in 'fatigue' as a function of the 'landing sequence' (when the effects of 'turbulence' were nullified). In the same way, we found a genuine increase in 'performance' and a decrease in 'stick activity' as a function of the 'landing sequence'. Our conclusion is that these changes reflect learning processes, and that the pilots change their techniques to cope with the landings. It is interesting to note that 'stick activity' changes over the landings, as it is often considered as a stable characteristic. The pilots of the studies are experienced, and their learning curves should have levelled out. However, the changes in the residuals for 'performance' and 'stick activity' as a function of 'landing sequence' seem to contradict this assumption, and the curves reflect an **accelerating** learning process?

In study one we found that the variables could be ordered in a sequence starting with the independent variable 'turbulence' and ending with the variable 'pilot induced oscillations'. We concluded that the sequence forms an embryo to a model of the internal causal relationships between important aspects of the landings.

In a first step we found that eight of the manifest or measured variables could be reduced to two latent factors: 'workload' and 'efficiency'. The reliability of the factors was established, and, furthermore, both are combinations of subjective ratings **and** objective measures, which support their validity.

The causal relationships between the independent variable "turbulence' and the two factors 'workload' and 'efficiency' were tested statistically by means of Structural Equation Modelling.

As can be seen from the model, increases in 'turbulence' are followed by increases in 'workload'. Increases in 'workload' are, in their turn, followed by decreases in 'efficiency'. Thus, 'workload' mediates the effects of 'turbulence' on 'efficiency'. From the plots of empirical data we found that low 'workload' predicts high 'efficiency', but also that high 'workload' not by necessity predicts low 'efficiency.' The variance in 'efficiency' increases as a function of 'workload'. The increased variance shows that the precision in the predictions of the pilots' 'performance' decreases, when the 'workload' increases.

From our regression analyses we found that 'aircraft handling qualities' and 'pilot induced oscillations' can be estimated from other variables as turbulence level and motions of the aircraft at touch down. The estimates are of practical interest, because they can be used as guides by systems developers.

General discussion and conclusions

One central objective of our studies was to analyse whether a moving base simulator could produce more realistic and useful cues (i.e., give the pilot better sensory feedback) than a fixed base simulator in simulations of turbulent approach/landings of high performance aircraft.

In the flight task analysed in the first study, we found that the pilot's 'stick inputs' were increased, when the motion system was engaged. We also found that the 'difficulty' level was higher and the 'performance' level lower under the same condition. Hence, in simulations without motion, important problems of the pilot - aircraft interaction in real flight will be underestimated, or one runs the risk of overestimating the 'handling qualities' of the real aircraft.

The pilot's acquisition of skill (i.e., his learning process) is based on his performance feedback. However, lack of relevant cues diminishes the pilot's possibilities to estimate his performance (i.e., to get performance feedback), and a positive transfer of training from simulations to real landings is less likely.

When comparing the matrices (i.e., the relations between the variables) we found a close correspondence between the 'motion' conditions of the two studies. The relations between 'performance' and the other variables were almost identical under the 'motion' conditions of the two studies. These relations are important, as the pilots' acquisition of skill is based on their performance feedback. Accordingly, the results of the second study validated the importance of motion in these types of simulations.

The interindividual differences in 'stick activity' in study one were also verified in the second study. Furthermore, there were differences in the pilots' increase in gain as a function of 'turbulence'. The pilots used the stick in different manners, like different people having different hand-writing. These differences are, of course, of central importance in control systems development. The systems must tackle the inter-individual differences. Could these interindividual differences in control behaviour be reduced by means of specific training, or are they manifest traits? In this study we found that the pilots' gain changed over time. Accordingly, the pilots' control behaviour has state characteristics and training might influence it.

Contrary to the pilots' control behaviour there were no differences between the pilots with respect to 'aircraft movements at touch down'. Accordingly, good landing performance may be achieved with different 'control behavior'.[23] On the other hand, perceived 'risk', 'induced oscillations', and 'mental workload' increased, and 'performance' (both subjective and objective), and 'aircraft handling qualities' decreased as a function of 'stick activity' or gain, when the 'turbulence' was held constant. Thus, there were genuine relations between gain and the other measures.

Our conclusions from the decreases in 'stick activity' and increases in 'performance' as a function of the 'landing sequence' are that these changes reflect learning processes, and that the pilots change their techniques to cope with the landings. These changes are interesting, as the pilots were experienced and their learning curves should have levelled out. Our conclusion is that the curves may reflect learning processes related to specific characteristics of the simulation. It should be noted that the curves reflect an accelerating learning process.

As can be seen from the model analyses, increases in 'turbulence' are followed by increases in 'workload', and increases in 'workload' are, in their turn, followed by decreases in 'efficiency'. Thus, 'workload' mediates the effects of 'turbulence' on 'efficiency'. We found that low 'workload' predicts high 'efficiency', but also that high 'workload' not by necessity predicts low 'efficiency'. The variance in 'efficiency' increases as a function of 'workload'. The increased variance shows that the precision in the predictions of the pilots' performance decreases when 'workload' increases. However, high

'efficiency' or performance in combination with high 'workload' (i.e., high mental effort) is more liable or sensitive to disturbances than high performance in combination with low 'workload'. The mental reserve capacity is reduced in the former case. Thus, the relation between 'workload' and 'efficiency' discloses how robust the pilot's landing performance is.

From psychometric and statistical points of view the factors 'workload' and 'efficiency' are powerful. It means that they can be used as practicable measures in systems evaluation and development. They can be used in, e.g., comparisons of different editions of flight control systems.

That we, to a substantial degree, could estimate or predict the pilots ratings of 'aircraft handling qualities' and 'pilot induced oscillations' is important. The estimates are of practical interest because they can be used as guides by systems developers as measures in the evaluation and development of electrical flight control systems (EFCS) of modern aircraft. Theoretically, it is of general interest to know the factors underlying pilot ratings.

References

1. Huddlestone, H. F. & Rolfe, J. S. (1971). Behavioral factors influencing the use of flight simulators for training, *Applied Ergonomics, 2*, 141-148.

2. Caro, P. W. (1977). Platform motion and simulator training effectiveness, *Proceedings of the 10th NTEC/-Industry Conference* NAVTRAEQUIPCEN IH-294. Orlando, Florida.

3. Caro, P. W. (1979). The relationship between flight simulator motion and training requirements, *Human Factors, 21*, 493-501.

4. Levison, W. H. & Junker, A. M. (1978). *A model for the pilot's use of motion cues in steady-state roll-axis tracking tasks,* AIAA paper no. 78-1593, Flight Simulation Technologies Conference, a Collection of Technical Papers, American Institute of Aeronautics and Austronautics. Arlington, TX , New York.

5. Puig, J. A., Harris, W. T. & Richard, G. L. (1978). *Motion in flight simulation: An annotated bibliography,* NAV-TRAEQUIPCEN IH-298, Naval Training Equipment Center, Orlando, Florida.

6. AGARD. (1980). *Fidelity of simulation for pilot training*, AGARD AR-159, Advisory Group on Aerospace Research and Development, Paris.

7. Rolfe, J. M. & Staples, K. J. (1986). *Flight simulation*. Cambridge University Press, Cambridge.

8. Stark, E. A. (1989). Simulation, in Jensen, R. S (ed), *Aviation psychology*, Gower Technical, Aldershot.

9. Scanlon, C. H. (1987). *Effects of motion cues during complex curved approach and landing tasks. A piloted simulation study,* NASA Technical Paper 2773, Langley Research Center, USA.

10. Repperberger, D. W. (1995). *Biodynamic and spacticity reduction in joystick control via force reflection*, Crew Systems Directorate and Biocommunications Division, Wright-Patterson AFB, Ohio.

11. Hosman, R. J. A. W. (1997). *Visual-Vestibular Interactions in the Perception and Control of the Aircraft Motions by the Pilot,* AIAA paper 97-3802, Modelling and Simulation Technologies Conference, New Orleans, USA.

12. Strachan, I. W. (1998). *To move or not to move?* That is the question, ITEC, 9th International Training & Education Conference, Lausanne, Switzerland.

13. Hosman, R. J. A. W., & van der Vaart, J. C. (1981).. Effects of vestibular and visual perception on task performance, *Acta Psychologica, 47*, 271-287.

14. van der Pal, J. (1996). Personal communication.

15. Svensson, E., Angelborg-Thanderz, M., Sjöberg, L. & Olsson, S. (1997). Information complexity - pilot performance and mental workload in combat aircraft. *Ergonomics, 40*, 362-380.

16. White, A. D., Hall, J. R. & Tomlinson, B. N. (1991). *Initial validation of an R&D simulator with large amplitude motion*, AGARD Conference Proceedings 513, Paris.

17. AGARD. (1995). *Flight vehicle integration. Panel vorkshop on pilot induced oscillations*, AGARD AR-335, Paris.

18. White, A. D. (1994). *Measures of flight simulator fidelity based on pilot control behaviour*, Flight Systems Department, Defence Research Agency, Bed-ford, UK.

19. Svensson, E., Angelborg-Thanderz M., Sjöberg, L. & Gillberg, M. (1988). Military flight experience and sympathoadrenal activity, *Aviation, Space, and Environmental Medicine, 59*, 411-416.

20. Angelborg-Thanderz, M. (1990). *Military flight training at a reasonable price and risk*, Economics Research Institute, Stockholm School of Economics and FOA report C 50083-5.1, Stockholm, Sweden.

21. Svensson, E., Angelborg-Thanderz, M. & Sjöberg, L. (1993a). Mission challenge, mental workload and performance in military aviation, *Aviation, Space, and Environmental Medicine, 64*, 985-991.

22. Svensson, E., Angelborg-Thanderz, M. & Sjöberg, L. (1993b). *Risk perception in military aviation*, Paper prepared for the conference Consequences of Risk Perception, 28-29 September, 1993, Center for Risk Research, Stockholm School of Economics, Sweden.

23. Ersson, J. (1998). *Characteristics of pilot control behaviour in high performance aircraft during simulated landings in turbulence*, Bachelor's thesis, University College of Mälardalen, Sweden.

24. Svensson , E., Angelborg-Thanderz, M. & Wilson, G. F. (1999). *Model of pilot performance for systems and mission evaluation - psychological and psychophysiological aspects*, Technical Report, USAF/RL.

25. Cooper, G. E. & Harper, R. Jr. (1969). *The use of pilot rating in the evaluations of aircraft handling qualities*, Report No. NASA-TN-D-5153, Moffett Field, CA, Ames Research Center, National Aeronautics and Space Administration Center.

26. Schiffrin, S. S., Reynolds, M. L. & Young, F. W. (1981). *Introduction to multidimensional scaling theory, methods, and applications*, Academic Press, New York.

27. Jöreskog, K.G., & Sörbom, D. (1984). *LISREL VI. Analysis of linear structural relationships by maximum likelyhood, instrumental variables, and least squares methods*, University of Uppsala, Sweden.

28. Jöreskog, K.G., & Sörbom, D. (1993). *LISREL 8: Struc tural Equation Modelling with the SIMPLIS Command Language*, Lawrence Erlbaum associates, Inc. Broadway.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37219 AIAA-2000-4077

STANDARD FLIGHT SIMULATOR AS A SPATIAL DISORIENTATION TRAINING DEVICE

Cezary Szczepański, Ph.D., M.Sc.Eng., Warsaw University of Technology, ETC-PZL Aerospace Industries, Poland
Col. F. Wojciech Skibniewski, Ph.D., M.Sc.Eng., Polish Air Force Institute of Aviation Medicine, Poland
Maj. Pil. Marek Gąsik, M.Sc.Eng., Polish Air Force Institute of Aviation Medicine, Poland
Lt. Col. Olaf Truszczyński, M.D., Polish Air Force Institute of Aviation Medicine, Poland

ABSTRACT

During flight the spatial disorientation may appear as a result of inadequate perception of the position or attitude of an aircraft in comparison of the co-ordinate system constituted by the Earth surface. It is extremely difficult to say how often spatial disorientation became the reason of military aircraft accidents [6] but in Poland it is calculated at around 8%.

Two kinds of experimental flights on the flight simulator were performed. The first one running in IFR and the second one in VFR conditions. The flights' profiles have included: take off, some navigational tasks, approach, final approach and landing. The valuation of flight performance was done on the basis of registration of eye movements and progress in following attempts of flights.

The obtained results indicated that in IFR conditions the symptoms of loss of spatial orientation were observed. In those cases the two types of reaction took place:
- type I of disorientation – no counteraction,
- type II of disorientation – counteraction.

Within the second type cases the oculograph registration indicated increasing intensity of eye saccadic movements and larger duration of fixation.

In good meteorological flight conditions there were not observed eye saccadic movements and duration of fixation significant changes. It means, that no spatial disorientation have appeared.

Additional conclusion of our research was, that method of eye saccadic movements measurements can be used for detecting the loss of spatial orientation by the pilots during the simulated and real flights.[1]

[1] Copyright © 2000 by Polish Air Force Institute of Aviation Medicine. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

1. INTRODUCTION

Problem of spatial disorientation is widely recognized as the most critical and one of the most serious reason of flight accidents. In our paper we would like to describe some procedures developed at the Polish Air Force Institute of Aviation Medicine with the use of flight simulator "Iapetus" (showed in Fig.1) to counteract the phenomenon of spatial disorientation including situational awareness and visual illusions. We refer to the possibility of standard flight simulator use for investigation of the spatial disorientation and training embracing that phenomenon.

Disorientation can be understood as a pilot reaction during the flight when he is forced to focus his attention on difficult meteorological conditions or the other part of the task, which doesn't allow him to control the whole flight situation. Pilots have to concentrate on different aspect of flight situation, i.e. speed, altitude, performed mission, that make them more susceptible to spatial disorientation. It may be the reason of being unaware disoriented (disorientation of type I). The pilot's flight performance in such a case is arisen because of perceptual misinformation. What is important pilot may fly unaware of his disorientation until it is too late for corrective action.

There are flight manoeuvres, which provoke spatial disorientation [8]. When pilot is conscious of such phenomenon then is possible to take up his earlier reaction and correction. In such a case pilot performs the manoeuvre being aware of perceptual and vestibular misinformation, which can be intellectually control through corrective information coming from the flight instruments and indicators. It was mentioned that spatial disorientation may occur in different phase of flight. Can special training counteract it? How does deal with it? These kinds of questions are usually asked by the pilots.

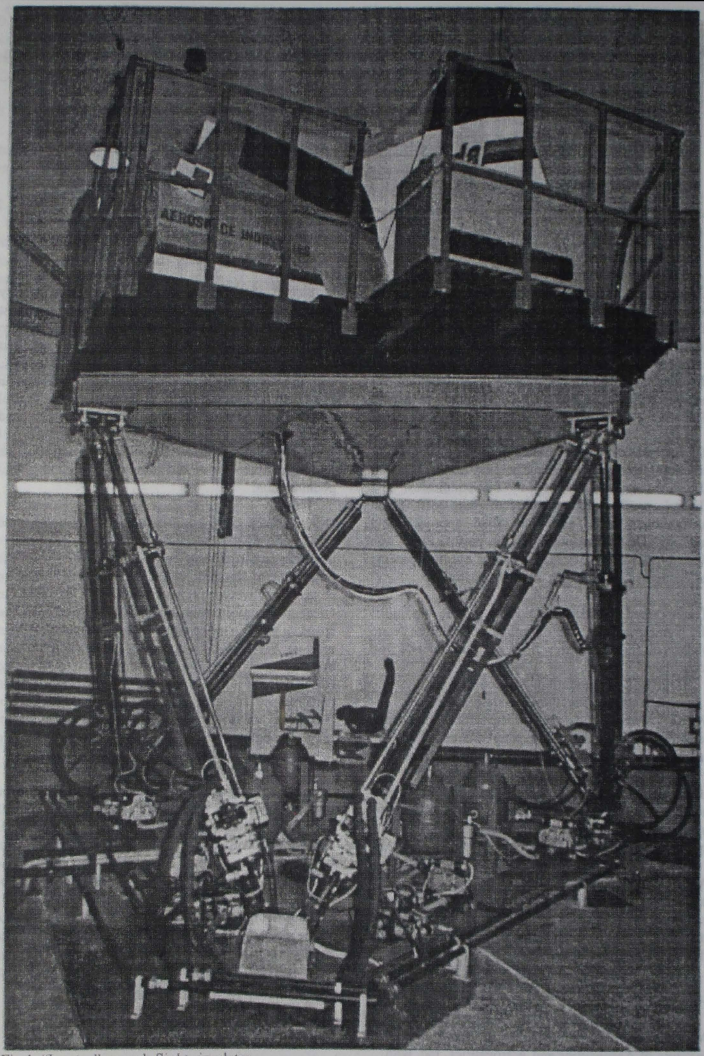


Fig.1. "Iapetus" research flight simulator.

For the purpose of that paper we have defined spatial disorientation as a state characterised by the false perception of one's position in relation to the Earth surface, which is caused by the senses misinterpreting the pilot's position in space. We have limited our paper only to the spatial disorientation caused by the visual factors.

The reason of such decision was, that during our research and analysis we found that classic hexapod (Stewart type) motion system of our flight simulator is practically unusable for spatial disorientation research and training. We shall present the discussion of that in the next paragraph.

For our practical purposes we will discuss only vision as the most important source of information, which provide orientation to pilots. In that manner vision can also be thought of as being the most important sense controlling flight situation. That visual dominance is a potential cause of illusions and disorientation during flight. Aircraft accidents have occurred because of illusions caused by limitations of our sense of vision. Knowledge of these illusions is an important step in the prevention of possible incidents and accidents. The reduction of aviation mishaps can be achieved through improvement of spatial disorientation understanding and development of pilot training programs.

2. HEXAPOD MOTION SYSTEM IN SPATIAL DISORIENTATION TRAINING

This paragraph contains a detailed technical appraisal of motion base possibilities for use in spatial disorientation training. That has been done on the base of our research and tests, and literature analysis, e.g. [5].

The primary purpose of a motion system of a spatial disorientation training device is to provide correct vestibular cueing to introduce spatial disorientation illusions in a realistic flight environment. To effectively accomplish this task, the motion system must have two very important characteristics:

- continuous yaw motion cueing with simultaneous pitch and roll cueing,
- subthreshold motion control capability.

"Iapetus" flight simulator motion system has the second characteristic and the first one we tried to achieve.

The six degrees of freedom hydraulic motion system provides transient cueing in pitch, roll, yaw, surge, sway and heave. However, it cannot provide the continuous yaw motion cueing necessary to support spatial disorientation training.

To only way to add to our flight simulator the continuous motion cueing was to incorporate a 360°, continuous motion yaw base. We realised, that careful attention must be given to where the 360°, continuous motion yaw base is integrated into the system. There are two options. The continuous yaw drive can be placed:

- underneath the hexapod motion base,
- on top of the hexapod motion base.

2.1. Yaw drive underneath the hexapod motion base

Placing the continuous yaw drive system under the hexapod motion base represents the only practical way to achieve simultaneous continuous yaw and pitch and/or roll. Unfortunately, in a hydraulic six degree of freedom system design, this is very difficult and potentially expensive configuration. Either the hydraulic pumps must be remotely located or they must be located on top of the yaw drive system. Neither of these approaches is feasible.

If the hydraulic pumps were remotely located a very sophisticated hydraulic rotating joint would be needed to supply hydraulic fluid to the actuators during 360° rotation. Even if such a joint exists, it would no doubt be both expensive and maintenance intensive. Accordingly, this approach is not practical.

If the hydraulic pumps were located on top of the hexapod motion base, tremendous amounts of electrical power would have to be passed through slip rings to power the pumps. Additionally, the pumps would be extremely noisy and since they were located in the training area (right underneath the trainee) they would interfere with training. Last, locating the hydraulic pumps on top of the continuous yaw motion system would add considerable weight that would need to be supported by the yaw bearing. Accordingly, this approach is not practical either.

2.2. Yaw drive on top of the hexapod motion base

This configuration disqualifies the device for spatial disorientation training because in this configuration, simultaneous continuous yaw plus pitch and roll are not possible.

Exact yaw position becomes extremely important if the yaw turntable is positioned on top of a hexapod motion base. In order to have simultaneous motion, the hexapod motion base would always have to know precise yaw position. Otherwise, pitch would turn into roll and back to pitch during yaw rotation (see Fig.2). The same thing would happen with roll. Co-ordination the instantaneous yaw drive position with the hexapod motion base would require very sophisticated motion, most likely impossible. Even if degree of control were possible, the pilot would

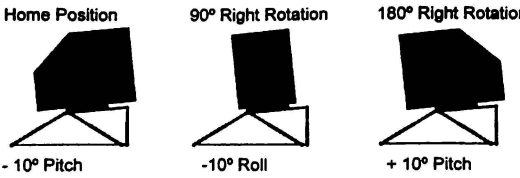


Fig.2. Limitations of a 6 DoF with 360° Yaw Drive on Top.

experience unacceptable transient motion cueing as the hexapod struggled to keep the pitch and/or roll vectors aligned while the yaw base was continuously rotating. The result of that deficient design is, that the only motion cue that can be simultaneously used with continuous yaw is heave. Alternately, if pitch and/or roll are needed only transient yaw can be introduced.

Since we have found, that simultaneous continuous yaw with pitch and/or roll is not possible in this configuration, the following types of illusions cannot be accurately created:

- Coriolis [3],
- Graveyard Spin,
- Graveyard Spiral,
- Leans,
- G-Excess.

Another words saying, the only vestibular illusion that this type of motion base can accurately and realistically support is the basic somatogyral illusion. And that capability is no better that the Barany Chair has, which is much simpler and cheaper.

2.3. Linear excursions effects

Introduction of linear excursions in sway and surge into the above described motion system, would introduce another serious issue during the yaw rotation, that being out-of-balance rotation. Excursions in sway or surge will drive the centre of mass of the device away from the centre of rotation. This motion will result in a dangerous off balance condition that could:

- introduce motion cues that could result in serious pilot injury,
- result even in catastrophic motion base failure.

Accordingly, linear excursions in sway and surge are not only unnecessary for spatial disorientation training but are actually undesirable.

Combining together the two technologies: a hexapod, hydraulic motion base and a yaw turntable, in order to get tool for research and training of spatial disorientation did not give us the desirable effect. Moreover, could create a motion system not only incapable of supporting spatial disorientation training but also that has the set of very undesirable characteristics. These include:

- incapability of supporting realistic spatial disorientation research and training for the most vestibular illusions,
  potentially unsafe system due to possibility of unbalanced rotation,
- very complex system,
- expensive to purchase,
- high operation and maintenance costs, making the cost per hour undesirable.

Concluding we can say, that combining a six degrees of freedom hydraulic motion system of standard flight simulator with the yaw turntable is not an appropriate solution for spatial disorientation research and training.

3. OTHER CHARACTERISTICS OF SPATIAL DISORIENTATION DEVICE

As we have mentioned in the previous paragraphs, our main goal was to extend the standard flight simulator scope of usage with the spatial disorientation training. We have discussed the hexapod motion system limitation for that purpose. Now we shall analyse the other characteristics of standard flight simulator as the spatial disorientation training device.

That kind of training device should complement academic training by presenting a realistic, interactive environment.

3.1. Environment realism

Environment realism is a critical quality of highly effective spatial disorientation training device. Lack of realism can result in poor learning and even negative training. The pilot who finds himself in less realistic environment may do things that he would not normally do in the actual aeroplane. Then we can get negative training. Additionally, in that condition

pilot may divert his attention from desired learning outcomes into critiquing the device. Then we get poor learning transfer with the pilot just "filling the square" so he can go out and fly the real aeroplane. Let us analyse the standard flight simulator characteristics from the necessary realism of environment point of view.

3.1.1. Closed cockpit environment

In our "Iapetus" flight simulator we have closed cockpit environment. It places the pilot "in the cockpit", which is very important. This sets the pilot's proper mindset for his training session.

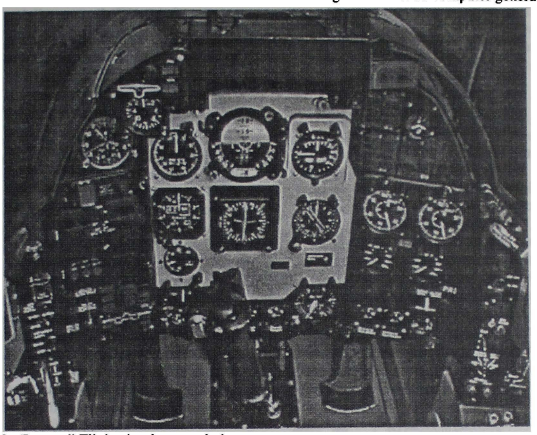


Fig.3. "Iapetus" Flight simulator cockpit.

3.1.2. High resolution, realistic visuals

"Iapetus" visualisation systems gives the pilot 120° horizontal and 28° vertical field of view, produced by the three channel collimating system with computer generated imagery. The resolution 1240 by 860 pixels per channel and high realism of presented pictures increase the reality sensed by the pilots. Since 80% of the information perceived by the pilot is through his eyes, the high quality out of the window display is critical to create a realistic training environment. In the standard modern flight simulator that requirement is fully fulfilled.

3.1.3. Flight controls

The flight controls in spatial disorientation training device should be located spatially correctly and should cause the reasonable aeroplane response when they are actuated. The above requirements were fulfilled more than minimally. We can say that in flight simulators they are better than necessary from the spatial disorientation training point of view.

3.1.4. Flight instruments

Flight instruments for the spatial disorientation training device can be computer generated or analogue ones. They should only respond correctly to engine and flight parameter changes. In the flight simulator that requirement is fulfilled from the beginning as one of the most important requirements (see Fig.3).

3.1.5. Model of flight dynamics

In a spatial disorientation training device a high quality model of flight dynamics, which responds aerodynamically correctly should be applied. In a standard flight simulator it is one of the most important requirement and is specified in the standards much more strictly and in detail. Flight simulator, which performs its standard requirements at that point, will be enough advanced for spatial disorientation training.

3.2. Environment interactivity

As it is commonly known, interactive training is absolutely necessary to learn psychomotor skills. We would not think of attempting to train a pilot to fly an aeroplane by only using classroom instruction and demonstration. In a traditional way of spatial disorientation training, however, that approach is being used. Not surprisingly, that approach has not been effective.

Interactivity starts with the inclusion of closed loop flight controls. These flight controls must provide a realistic response to the pilot's inputs. Within the interactive, effective spatial disorientation training, pilot must be allowed to fly the virtual aeroplane into the episode, recognise the spatial disorientation situation, change the priorities of his cockpit tasks and successfully resolve the spatial disorientation episode.

The spatial disorientation training device should also be equipped with the instructor station, which allows him not only to select and run previously stored spatial disorientation profiles, but also to introduce inputs while the scenario is running. That creates the possibility for unique multitasked learning environments. The above is naturally a part of standard flight simulator set and a way of training. So with that point standard flight simulator is ready to be a spatial disorientation training device.

A device for training the spatial disorientation should allow for recording the training session for its effective debrief [5,7]. Information learned during debriefings is invaluable for preventing spatial disorientation episodes since that information can be called upon, when a pilot makes inflight decisions during the perception-decision-action process. Interactivity is an important consideration throughout the pilot's spatial disorientation learning experience and should extend through preflight, inflight and postflight, during the training session. For that requirement fulfilment every standard flight simulator is ready, as it is required from any flight simulator.

So, we can conclude, that reality and interactivity environment of a standard flight simulator is good enough to use it as a spatial disorientation training device.

4. EXISTING FLIGHT SIMULATOR CHANGING

As it can be seen from the previously presented analysis of standard flight simulator configuration versus spatial disorientation training device, "Iapetus" flight simulator hardware should not have to be changed. The main scope of work for adaptation of that simulator to spatial disorientation training was in new software development. The following changes in the "Iapetus" software have been done:

- introduction of some additional elements into the visualisation software. They are mainly connected to the meteorological conditions of simulated flights;
- textures of Baltic Sea with its coast line;
- lights of cities with respect of their location. They are situated in vicinity of airfields with regard of their geographical co-ordinations. The most crucial are these, which accidental configuration of their street lights are in straight lines;
- textures of sky including star lights. They are presented as points, which are motionless but they are accidentally arranged and have diverse intensity of shining;
- upper limit of clouds level, which is diagonal to natural horizon line at angle of the range 1° to 5°;

  the aeroplane – target, which serves as a tracked object for the trained pilot for shifting his attention. The control of the aeroplane – target is being performed from the flight simulator instructor station. It also can be controlled automatically by the special software installed in the "Iapetus" software system. Then it has possibility of movement within the altitudes from 50 meters to 3000 meters, and has flight dynamics characteristics of real TS-11 "Iskra" jet trainer aeroplane. That aeroplane – target appears in the central channel of simulator visualisation system for being better pilot's attention shifter;
- "freezing" the flight instruments in their actual position with possibility of return to the normal mode after time defined by the instructor/researcher.

The introduced changes allowed for simulating the spatial disorientation illusions coming to the pilot through his sense of vision [2].

5. EXPERIMENT METHODOLOGY

We have decided to develop an experimental methodology of spatial disorientation training with the use of "Iapetus" flight simulator.

The initial course was designed for five military pilots. It has included theoretical course on spatial orientation and explanation on the experimental flights, which have to be performed. In the first part of the course pilots have been introduced into the spatial orientation background, i.e. they were informed when the spatial disorientation may appear and what are its consequences. The main goal of that phase of training was deepening pilots' knowledge on the phenomenon and understanding mechanisms

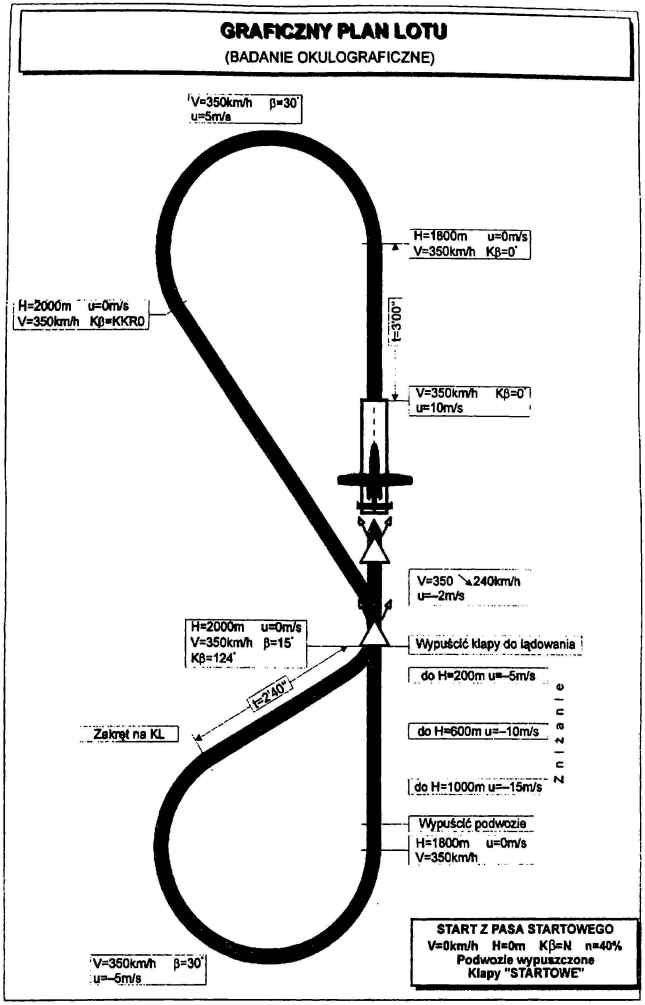


Fig.4. Experimental flights profile.

leading to spatial disorientation. The second part of theoretical preparation was devoted to explanation of meaning and goals of experimental flights, which have to be performed by the pilot in the flight simulator. The pilots were prepared for not only to be the subjects but also the test pilots evaluating the same flight profile as a spatial disorientation demonstrator.

There were prepared two experimental flights, which profile is presented in Fig.4. The flight profile consists of: take off, climbing, a few course changes, approach, final approach and landing according to NDB (non directional beacon) system. The first flight runs in IFR (Instrumental Flight Requirements) conditions with the strong turbulence presence. The second one goes on without turbulence and in VFR (Visual Flight Requirements) conditions.

Two methods of flight evaluation were used. The first, based on the standard evaluation of flight task performance, as a deviation from the required flight parameters. And the second one, where the eye movements were registered with the use of oculograph.

6. RESULTS DISCUSSION

Among the first experimental type of flights there were two without space disorientation observed and three where that phenomenon took place. In the flights with disorientation observed, there were two kinds of pilot reaction:

- type I : no counter reaction (1 case),
- type II: counter reaction observed (2 cases).

In the last two the oculographic registrations indicated increasing intensity of eye saccadic movements and longer duration of fixation, which was probably caused by the longer time necessary for verification of flight parameters. It was not observed in type I disorientation.

For the flights took part in VFR conditions there were not observed any significant changes in eye saccadic movements and no extension of eye fixation duration . That can be a confirmation of non existing spatial disorientation during these flights.

6. CONCLUSIONS AND CLOSING REMARKS

It is necessary to conduct the extensive tests allowing for evaluation and adoption to use the standard flight simulator as a spatial disorientation training device.

On the basis of performed tests and analysis we can say, that training of vestibular illusions causing spatial disorientation in a standard flight simulator with hexapod motion system (Stewart type) is not feasible.

The standard flight simulator is very well prepared for training pilots in spatial disorientation caused by the visual illusions [9]. Only some additions to the simulator software is necessary.

The oculograpic saccadic eye movement measuring method can be useful for spatial disorientation detection during the flight in real aeroplanes and also in flight simulators. Particularly for the highly maneuverable aeroplanes that method is seemed to be useful. The measuring equipment is comparatively light and compact and does not disturb substantially the pilot's field of view.

It is also concluded that is reasonable to introduce standard procedures based on the "Iapetus" flight simulator and treated as an integrated element of counteracting spatial disorientation training system [4].

REFERENCES

[1]Benson, A.J. and Burchard, E.: Spatial Disorientation in Flight – A Handbook for Aircrew. AGARD-AG-170, NATO/AGARD, Neuilly-sur-Seine, France, 1973.

[2]Braithwaite M., G., Hudhens J., J., Estrada A., Alvarez E., A. An Evaluation of the British Army Spatial Disorientation Sortie in U.S. Army Aviation. Av.Space Environ. Med. 69 (8): 727 – 32, 1998

[3]Gillingham, K.K.: Some notes on the treshold of the vestibular Coriolis effect and its significance to aircrew. SAM-TR-65-55. USAF School of Aerospace Medicine, Brooks Air Force Base, TX, 1965.

[4]Johnson P.A., Estrada A., Braithwaite M., G., Manning J., C. Assessment of Simulated Spatial Disorientation Scenarios in Training U.S. Army Aviators. RTO, Human Factors & Medicine Panel, San Diego, CA, 12 –22.10.1998 r.

[5]Leland R.A., Requirements For An Effective, Low Cost Spatial Disorientation Trainer. AeroMedical Training Institute, Philadelphia, PA, 2000 (Interim Report).

[6]Nuttall, J.B. and Sanford, W.G.: Spatial Disorientation in Operational Flying. Publication M-27-56. United States Air Force Directorate of Flight Safety Research, Norton Air Force Base, CA, Sep. 12. 1956.

[7]Orientation/Disorientation Training of Flying Personnel: A Working Group Report. AGARD-R-625. Edited by A.J. Benson, NATO, Neuilly-sur-Seine, France 1974.

[8]Peters, R.A.: Dynamics of the vestibular system and their relation to motion perception, spatial disorientation, and illusions. NASA-CR-1309. National. Aeronautics and Space Administration, Washington, DC, 1969.

[9]Pitts, D.G.: Visual illusions and aircraft accidents. SAM-TR-67-28. USAF School of Aerospace Medicine, Brooks Air Force Base, TX, 1967.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37220

AIAA-2000-4080

LOW-COST SIMULATION SOLUTIONS FOR AEROSPACE PROJECTS

Ferdinand Cornelissen, Adrick Broeils, Leon Bremer and Leslie Gale.
Origin Nederland B.V., Technical Automation / Advanced Technology Solutions,
Bakenmonde 2, P.O. Box 1444, 3430 BK Nieuwegein, The Netherlands
Tel: (+31)(0)30 - 608 8888
Fax: (+31)(0)30 - 606 0577
Email: {ferdinand.cornelissen, adrick.broeils, leon.bremer, leslie.gale}@nl.origin-it.com
Url: http://www.origin-it.com

Abstract

Market trends are that everyone wants their simulation solutions to be low-cost. With increasing capabilities of consumer market computer hardware, Origin has investigated the technical possibilities of creating a low-cost simulation environment. This resulted in a generic architecture for simulation solutions. The architecture is based on the logical distribution of components, which realizes among others hardware-in-the-loop, man-in-the-loop, 3D visualization, real-time systems and distributed simulation. Several research projects have resulted in a set of tools for developing a distributed simulation environment. Recently we added wide-area network (WAN) connectivity to our architecture during a project called "EPOS Remote Control". Using this interface it is possible to conduct simulation runs with hardware in the loop (HIL) over long distances. Such a solution is especially interesting for collaborations between individual companies and institutes that are spread over a large area. The connectivity technology we have used is ISDN, a low cost solution for point-to-point (PPP) connections. We conclude that wide-area network connectivity has been successfully imported in the simulation environment as indicated by results of several test scenarios run on the European Proximity Operations Simulator (EPOS).

Introduction

This paper focuses on the aspects of simulation that we have addressed using a low-cost approach. This approach is based on an architecture for distributed simulation systems as presented by Bremer et al[1]. Results of several projects regarding low-cost simulation solutions in the Space Industry are presented. Our latest project, called "Remote Control", tackled the problem of connecting simulation subsystems over large physical distances using a low-cost connectivity technology. The increasing Quality of Services versus the costs of these services has led to the situation where low-cost ISDN technology might be usable for network connectivity in a wide-area distributed simulation environment. In this paper we describe the Remote Control (RemCon) project in the context of Origin's distributed simulation architecture and implementations based on the architecture using a low-cost approach. Results of this particular issue and the caveats and pitfalls we encountered are presented as well.

The outline of this paper is as follows. First we will illustrate the simulation architecture as envisaged by Origin, followed by the results of a number of low-cost implementations based on this architecture. We then will proceed by reporting on our findings regarding the project "Remote Control", where low-cost wide-area connectivity is used in a distributed simulation. Our conclusions will be presented in the last part of this paper.

Low-Cost Simulation

The remote control project is part of our ongoing development of distributed simulation solutions. Responding to market trends and the fast-paced technological advance of low-cost consumer hardware, Origin has focussed on a cost-effective approach to developing simulation solutions. This resulted in a generic simulation architecture based on a distribution of logical components of a simulation environment. Clearly, to support the distribution of these logical components, some form of connectivity is needed. Up till now, different technologies have been used, ranging from reflective memory boards to local-area networks (LANs) based on Ethernet. The objective of the Remote Control project is to extend this range of technologies to wide-area networks (WANs) to support wide-area simulation environments. In the following section we will

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

present the distributed simulation architecture to provide the background for the remote control project.

Distributed Simulation Architecture

To support a wide spread applicability of low-cost simulation technology, and to support the requirements of developers, Origin has been faced with the challenge to define an architecture that supports *re-usability* (not targeted to a specific context), *extendibility* (adding more functionality) and *scalability* (growth in performance). The simulation architecture proposed by Origin is generic, as it separates context, defined by the models, from simulation features. It is extendible

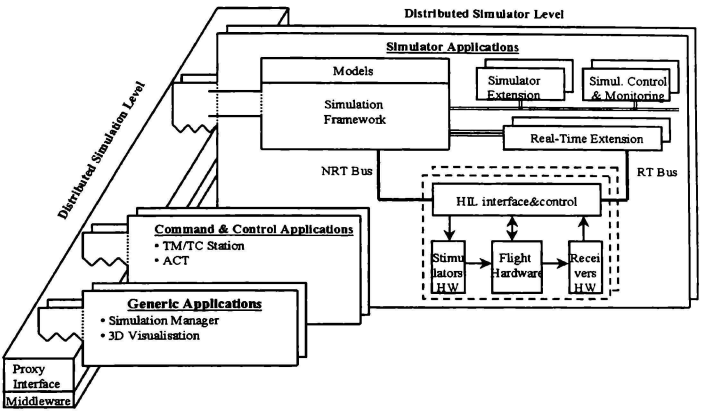


Figure 1. Origin's low-cost distributed simulation architecture.

by allowing users to select only initially required components and extending their facility with additional components when necessary. It is also scalable as the decomposition is on a logical level, leaving the selection of platforms a design issue within the implementation of a specific simulation facility.

The architecture as shown in figure 1 contains two complementary levels of simulation. On the one hand, the coherence of the components is based on a distributed simulat*ion* level, while, on the other hand, the coherence of components is based on a distributed simulat*or* level.

Simulation Level

At the simulat*ion* level, the components in the simulated world are communicating in the simulated domain. The components usually have a meaning in the simulated world as they represent satellites, command centers or ground-stations, etc. This means that simulated objects have knowledge of the simulation domain, and behave accordingly. A message-oriented protocol ensures the synchronization between the components. For instance, in a tool for Satellite Constellations planning[2] all simulated satellites were implemented as individual components, which could interact with each other by sending messages, reflecting their state, current knowledge and intentions. Thus, the communication protocol is part of the simulation, and defines the coherence between the components. Some simulations require a stronger coherence in contrast with other simulations. In some cases, inter-component coherence could even be subject of study within the simulation domain.

On the simulation level, the simulation domain is mapped onto a proxy layer. This proxy layer is build on top of a middleware layer which is responsible for the low-level distribution characteristics such as message queuing, synchronizing simulators, etc. The proxy layer

simplifies interfacing between simulators as most of the distributed character is hidden from the simulation applications. Typical component types on the simulation level are:

- *Simulator Applications:* these are applications that simulate the behavior of an object within the simulated world, regularly updating their state, monitoring the state of other objects and interacting with other objects. An example is a satellite simulator within a satellite constellation. Its internal construction is hidden from other satellites, but it publishes information when interacting with other satellites;
- *Command & Control Applications:* these are applications that do not represent an object in the simulated world but can interact with objects. A typical example is a telecommand and telemetry station, which receives information from a satellite and can uplink new commands, but is not an object with a representation in the simulated world.
- *Generic Applications:* Tools that can be used within every distributed simulation. Typically these respond to control and monitoring interfaces. An example is a 3D visualization tool, which can visualize the state and interactions of objects within the simulated world.

Simulator Level

At the simulat*or* level, the components within *one* simulator are communicating on a lower level. Here, knowledge of the *simulated world* is no longer an issue, rather than having detailed knowledge of each other regarding implementation or design. At this level, several components taken together are responsible for the functionality of one simulator. The distribution of the components can be a consequence of timing requirements, interfacing to hardware-in-the-loop, or man-in-the-loop, or by the need for raw processing power. Coherence between these components is high, as communication is constrained by real-time requirements, low-level hardware interfacing etc. A summary of typical components at the simulator level is presented here. A difference is made between 'models', which are context specific, and 'simulation framework', the generic part of a simulator that provides services to the models and the user of the simulator. As models are solution-specific, they will not be discussed. Apart from them the following simulator components have been defined:

- *Simulator Framework:* The simulator framework provides the model software with simulation services through a simulation framework API, and provides the user with services for simulator control, monitoring and logging. In addition it provides interfaces to the distributed simulation level, to generic non real-time simulator extensions, to generic real-time simulator extensions and HIL systems. Note that the simulator framework only supports non real-time execution of model software, i.e. only the order in which models are executed can be defined, not the timing at which execution starts.
- *Real-time extensions* support execution of 'tasks' according to (hard) real-time constraints. It is a generic extension, as tasks do not have a one-to-one relation to models. The component's objective is usually to achieve hard real-time behavior (the system shall meet deadlines). A less restrictive mode, called soft real-time (the system should meet deadlines), is also supported for those cases where the selected hardware is not capable of hard real-time systems support.
- *Simulator extensions.* Using middleware, non real-time parts of the simulator can either run on the same platform as the simulator framework or on additional connected platforms.
- *Simulator control and monitor interfaces.* Based on the middleware, simulator control and monitoring interfaces can either run on the same platform as the simulator framework or on connected platforms.
- *HIL interface and control system.* To control the stimulator and receiver hardware that are part of the simulation facility, a component is required that handles the interface between the simulator and this test facility hardware. This interface translates commands from the simulated domain to the test hardware domain and supplies monitoring capability to control and monitor the operation of the facility. Examples of required functionalities are user control and logging, interpolation / extrapolation on received commands and transformation of high-level commands to detailed hard real-time control of connected hardware. The interface component may vary considerably in complexity depending on the connected hardware. In addition it may be used in a stand-alone mode for open-loop testing of flight hardware. In the architecture two Hardware-In-the-Loop busses are defined: a Non real-time (NRT) and a real-time (RT) bus. For real-time busses the timing relation between all connected systems is known in contrast to non real-time busses.

In a simulation environment, both the simulat*or* components and simulat*ion* components are represented. However, these levels of simulation

are complementary, i.e. simulator components have no knowledge of the existence of simulation components and vice versa. In such a combination of both levels, the difference in coherence becomes clear, as components on the simulation level do not necessarily know of each other's existence, whereas the simulator components depend on each other.

Applications

To illustrate the generic architecture, we will present some of the applications of Origin's distributed simulation architecture. In these applications we were able to test our distributed architecture and its individual components. As a result of these projects, Origin has developed a set of tools, which are reused in ongoing simulation projects. In the projects presented here we developed tools for distributed simulation, real-time simulation, 3D visualization and hardware-in-the-loop.

ADS: A proxy layer for distributed simulation

During a research project on distributed simulation called "Advanced Distributed Simulation", a proxy layer to provide an interface to components at the simulation level was developed. The proxy layer creates a representative of each simulated object in the environment. This proxy layer is build on top of a middleware layer, which is responsible for the low-level functionality. The chosen technology of the middleware layer is the High Level Architecture (HLA) developed by the American Department of Defense. However, the proxy layer hides the middleware layer to the simulator applications, creating a simulation environment that is independent of the middleware. The proxy layer tool has been used among other things in projects regarding simulated constellations of satellites and distributed planning.

Satellite Constellations: distributed planning

The aim of the "Satellite Constellations" project is to build a tool to support a feasibility study of distributed planning in a constellation of autonomous smallsats[2]. The planning tool was implemented using the proxy layer that was developed during the ADS project. Using a variable sized set of satellites and ground-stations the proxy layer was tested for scalability. As was pointed out by Broeils et al.[3], it turned out that the proxy layer is a robust solution for scaleable distributed simulations.

VE: Virtual environments & simulation

Within any distributed simulation, a general 3D presentation of the simulated world is required, which allows inspection of the virtual world without influencing it. Such a tool is often referred to as a 'stealth'.

Based on the object-oriented proxy interface and an onward development in creating an object-oriented Virtual Environment, a low-cost 3D stealth has been created as a result of projects regarding "Virtual Environments & Simulation". The stealth supports the dynamic character of distributed simulations, where multiple instances (or federates

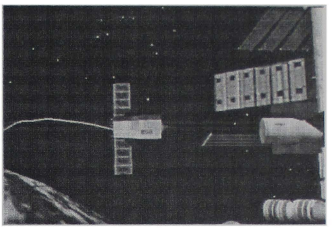


Figure 2. Visualization of the docking between the ATV and the ISS.

in HLA terminology) of a type of object can exist. Each object automatically receives a connected camera, orientation axes, head-up display, trail facility, interaction display and message display facility.

We have used the visualization tool in a large number of projects. Figure 2 shows a visualization of the docking between the Autonomous Transfer Vehicle and the International Space Station. Here, the visualization tool was applied in the context of the ELS project. However, the generic quality of the tool has made it possible to connect it to virtually any simulation environment based on the ADS proxy layer.

RTSIMNT: Soft real-time simulation

EuroSim is used widely as the real-time simulation environment in ESA projects. Origin is one of the consortium partners developing EuroSim with the primary responsibility for EuroSim's real-time scheduling facilities. In the project "Real-Time SIMulation on windows NT" (RTSIMNT), a first step has been undertaken by Origin to provide a low-cost EuroSim version running on PCs with the Microsoft Windows NT operating system. This

environment combines a simulator framework and a real-time extension as identified in the low-cost simulation architecture, on one platform. The RTSIMNT framework is a prototype of a complete low-cost simulator environment containing the simulation execution environment including real-time execution, as well as a graphical simulator development environment. The RTSIMNT environment features compatibility with EuroSim with respect to scheduling facilities, and is capable of soft-real-time simulations. The RTSIMNT framework has a limited support for external simulator extensions, external simulator control, and HIL interfaces. The implementation of EuroSim NT shall deliver an EuroSim version on Windows NT that covers all of EuroSim's features.

ELS: Real time communication

An evaluation between simulations on low-end and high-end platforms is assessed in the project "Evaluation of Low Cost simulations". The project was based on the Test & Verification Facility (TVF). This is a high-end implementation of a testbed for rendezvous and docking (RVD) scenarios between an automated spacecraft (ATV) and the international space station (ISS). A low cost version of the TVF was implemented using RTSIMNT for the model framework that was connected with the onboard computer simulator of the ATV using a military standard real-time bus. The aim of the project was to evaluate the possible performance differences between the high-end and the low-end implementation of TVF. As pointed out by Trompert and Keijzer[4], results indicated that no significant performance loss was measured. This means that a significant improvement of price / performance ratio is viable using a low-cost approach. Furthermore, in the scope of this paper, the ELS project yielded a real-time communication technology for intra-simulator component distribution using a military standard real-time bus (MIL-1553B).

Low-Cost HIL: A HIL interface

The purpose of the "Low-cost Hardware-in-the-Loop" project was to implement a low-cost interface for commanding and controlling a robotic system. The project has been carried out in collaboration with the German centre for aviation and aerospace (DLR). This center is home of the European Proximity Operations Simulator (EPOS), a hard real-time system designed for validation and verification of space rendezvous and docking equipment and subsystems. The real-time communication between simulator subsystems in this case consisted of a 'shared memory' segment, which is accessible by different systems using a reflective memory card. As presented by Schreutelkamp and Heimbold[5] results indicated that a low cost real-time system built according to a hybrid architecture and real-time communication using shared memory, is very well capable of replacing a high-end system at a fraction of the cost and without performance loss. The technology

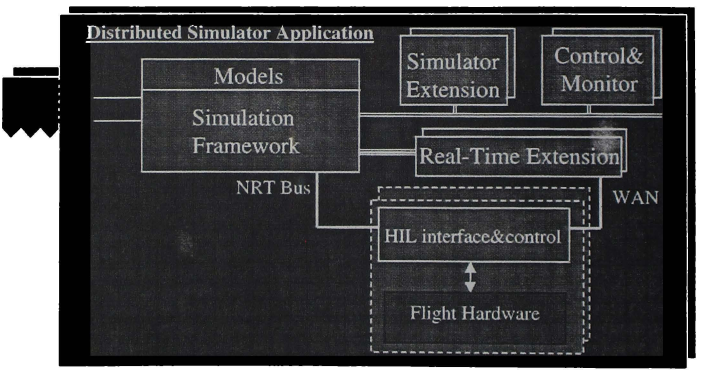


Figure 3. The WAN interface within a simulator application.

developed in this project therefore became the foundation on which a generic connection between the simulator framework and several real-time extensions will be developed.

Remote Control

As pointed out in the previous sections, by carrying out research projects on all aspects of the distributed simulation architecture concept, Origin was able to develop a set of tools and technologies that support a simulation environment. As distribution is a key concept in the architecture, several communication technologies have been examined. This includes real-time communication using reflective memory, military standard RT busses, dedicated HIL interfaces, and non real-time communication using LAN technology. The goal of remote control is to add low-cost WAN technology to this list of connectivity techniques.

In terms of our architecture, the remote control project examines a WAN interface between the real-time extension and hardware-in-the-loop interface & control component at the simulator level as depicted in figure 3.

Definition

The German laboratory for aerospace technology (DLR) facilitates the European Proximity Operations Simulator (EPOS). The EPOS facility has been developed in order to study and develop systems that realize a docking between two space objects, such as an Autonomous Transfer Vehicle (ATV) and the International Space Station (ISS). The EPOS simulator consists of a robotic system of flight hardware, which can be controlled and monitored by a set of computer systems. Figure 4 presents a photograph of the robotic system. Currently the hardware and the computer systems transfer hard real-time data using the existing local area network.

With the introduction of guaranteed quality of services over long distance by commercial network providers it becomes worthwhile to consider the usage of test beds, requiring data transfer in (hard) real time in a wide area environment. The expected benefits of such a configuration are an increased flexibility in the usage of such a test bed and essential cost savings by removing the necessity to have all required items and personnel at the same location. An excellent example of a test bed of this kind is the EPOS facility for rendezvous test campaigns. In order to reduce cost and extend the possibilities of the EPOS facility it should be possible to control and monitor the hardware in a distributed way, such that long distance communication between hardware stationed at physically remote places becomes possible.

The Remote Control project was initiated by the European Space Agency (ESA) to study the

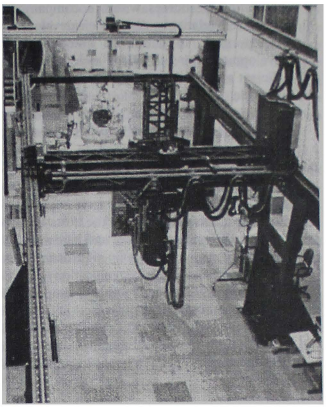


Figure 4. The EPOS Robotic System at DLR.

feasibility of controlling the EPOS hardware over long distances with the addition of live human feedback in digital audio and digital video format. This requires hard real time data and bulk data to be transferred over a wide area network and thus it provides an excellent test framework where issues such as wide area network (hard) real-time data transfer, collaborative engineering, etc. can be assessed.

Wide Area Distribution of Components

The distribution of components at the simulator or simulation level as defined by the distributed simulation architecture is realized by technologies such as local area networks, reflective memory cards, military standard real-time buses, etc. However these require that all components are implemented on machines which share a relatively small physical space, such as one floor, or one building. In most cases this will be sufficient for a given simulation environment. However, as is the case with most projects in the space industry, projects tend to be executed by consortia of institutes, companies and agencies. In most cases,

these partners are scattered over a wide area on the world. Development of a simulation environment or conducting simulations will involve persons who are located around the globe. In such cases, having the possibility to distribute the components of a simulation environment over wide areas will have benefits to all who are involved. Direct benefits of such a wide area distribution are:

- *Flexibility:* Resources within a company can be utilized relatively easy when development is performed 'in-house', as compared with development 'on-site'. This accounts for an efficient and flexible deployment of the project life cycle.
- *Ease of collaboration:* no physical relocation of personnel and hardware is necessary, because long distances need to be bridged purely by digital data. This will relieve the demands that are put on personnel and the project life cycle.
- *Reducing Costs:* Apart from the costs that are needed to relocate personnel and hardware, the project life cycle can take up significantly less time when the need of relocation can be omitted. Physical relocation has a deep impact on the planning of the project and is not without financial risk. When at the scheduled time of relocation not all deadlines have been met, this could mean a significant financial loss.

Based on these assumptions we have investigated the possibility of extending our architecture with wide area interfaces. Possible caveats that accompany such interfaces are related to:

- *Introduced Delay:* By using wide area connectivity, it is inevitable that some delay is introduced.
- *Long distances increase jitter:* Data which is send over long distances is subject to increased jitter if rerouting of datagrams is used. This means that the introduced time delay does not have a constant character, but the delay varies over time.

Simulation Environment

We have built a remote controllable setup based on the Test & Verification Facility (TVF). In figure 5 the original configuration of the TVF is depicted. The following components are identified:

- *Model Software (MSW):* This is the EuroSim TVF Simulator, which is the core of the simulation. It provides the mapping between actuators and sensors. It consists of the mathematical models of all sensors and the environment of the transfer vehicle. The MSW provides basic input for the onboard computer.
- *Onboard Computer Simulator (OCS):* The OCS simulates the onboard computer of the transfer vehicle. It is responsible for driving the thrusters, based on the measurements of all sensors.
- *Motion Operator Control (MOC):* This part of the simulation environment provides an interface to the hardware-in-the-loop. It controls the robotic system by sending motion primitives to the LDEC.
- *Laboratory Dynamic Environment Control (LDEC):* The LDEC consists of software that checks for inconsistencies in the set of motion primitives and the actual drivers to the hardware. We present the LDEC as one component here, including the robotic system, but it actually consists of a set of components.

In the original TVF configuration the link between the MSW and the MOC was established using LAN connectivity. The MSW is a synchronous

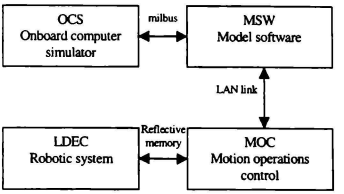


Figure 5. Original TVF setup.

command source for the MOC. It provides the simulated movement of the transfer vehicle, which is interpreted by the MOC, which eventually drives the physical robotic system. Originally there did not exist means of communication between the MOC operator and the MSW operator, other than the fact that they shared the same physical space. Thus communication occurred face to face.

In the Remote Control project the following components were added:

- *Relocated Sensor Model:* In order to test (hard) real-time data transfer, one crucial model was lifted out of the original model software and relocated at the motion operator control. This way we can simulate having a real hardware sensor available at the LDEC. The chosen sensor model simulates the Rendezvous Sensor. This sensor is responsible for measuring the position of the docking ports in the final stage of the docking. The onboard computer to accurately steer the transfer

vehicle to the ISS docking port uses measurements of this sensor. The sensor is crucial for docking, as wrong measurements surely lead to failure during the docking phase.

- *WAN interface:* The existing LAN interface is replaced by a WAN interface utilizing a TCP/IP protocol. Such an interface is added on both the MOC system, and the MSW system. The interfaces are responsible for setting up a WAN connection and transferring hard real-time data over this connection. Data from the MSW to the MOC represents motion commands to drive the robotic system, while data from the MOC to the MSW represents the measurements of the relocated sensor model.
- *Communication link between operators:* Because of the fact that both operators no longer share the same physical space, a communication link is added between the MOC operator and the MSW operator to support collaborative engineering.

In this new setup, two types of information are transferred over a WAN connection. First, information with highest priority is the hard real-time data that is transferred between the MSW and the MOC using the WAN interface. This data is hard real-time, as the actual robotic system is driven. Within one cycle of the real-time simulation, information has to be send back and forth between the MSW and the MOC. If this demand cannot be met, the components of the simulation environment will not be synchronized anymore, and a docking failure is inevitable.

Second, audio and video information is the other type of information that is transferred over a wide

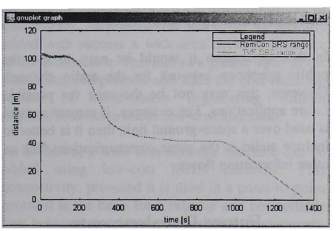


Figure 6. SRS sensor measured range during an original TVF run and a remote controlled TVF run (closed loop).

area connection. Although safety is the most important driver behind collaborative engineering, this information has a lower priority in contrast with the hard real-time data. The time window in which a hazardous situation may be detected and nullified is much greater than the time window in which the real-time data has to be transferred back and forth.

All communication is done using ISDN technology. This is a low cost technology yielding reliable data transfer rate. The bandwidth of the communication link is produced by 8 ISDN B channels, each providing 64 kbit/s. Most of this bandwidth will be allocated to the collaborative engineering data. Only one channel will be used for the hard real time data transfer, as the required amount of real-time data is < 32 kbit/s. The remainder of the bandwidth is used for live video and audio feedback between the operators.

The connection between the remote commanding site and the EPOS site is established using a point-to-point protocol (PPP). This effectively means that a 'direct' link is established between two points, in contrast with the internet protocol (IP) where a path in the network connectivity graph between two points will vary over time. The advantages of PPP over IP are twofold. First, using PPP guarantees a path with a constant characteristic in terms of packet roundtrip time and the absence of net congestion. Secondly, as a result of the 'direct' link that is established, PPP is more secure compared with the Internet protocol.

Testing Real Time Data Transfer

We have tested the WAN configuration using two different scenarios:

- *Open Loop Simulation:* In this scenario, the sensor model is fed with data as provided by the model software. This means that the information that is transferred over the WAN connection includes all necessary data to calculate the rendezvous sensor model. The simulated measurements of this sensor are then fed back over the WAN connection to the model software.
- *Closed Loop Simulation:* In a closed loop simulation, we feed the sensor model with actual measurements from the robotic hardware. We have chosen to feed a model with measurements instead of using a 'real' hardware sensor to limit the number of variations between the original setup and the remote control setup. Using a real hardware sensor adds many alterations to the setup and falls out of the scope of the project.

In both scenarios we monitored all variables of the transfer vehicle, such as x-, y-, z-position, velocities, fuel usage etc. and compared these to the results of runs on the original setup. Variables

of the sensor that are measured are the range to the target and the bearing of the line of sight to the target. Furthermore, we defined that for a test run to be successful the transfer vehicle must dock with

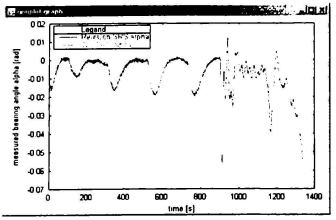


Figure 7. SRS sensor measured bearing angle alpha during an original TVF run and a remote controlled TVF run (closed loop).

the ISS. We conducted a series of runs, the results of which are presented in the following section.

Results

Comparing the monitored data between runs with the original setup and runs with the remote control setup, we conclude that the differences are insignificant. This is valid for all scenarios, including open loop and closed loop simulation runs. Figure 6 shows a typical plot of the combined measured SRS range of the transfer vehicle during an original run and a closed loop remote control run. Figure 7 and 8 show the measured SRS bearing angles during a closed loop remote control run and an original TVF run. As can be concluded from these plot-diagrams, differences between the sensor measurements during a remote controlled

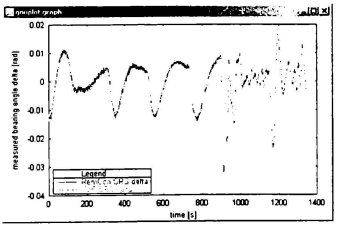


Figure 8. SRS sensor measured bearing angle delta during an original TVF run and a remote controlled TVF run (closed loop).

closed loop run and original TVF runs are insignificant. Docking is in both cases successful. We conclude that the remote control setup using WAN connectivity has been implemented with success.

Testing Collaborative Engineering

As stated before, collaborative engineering in the context of remote control, consists of live audio and video feedback between the operators at the remote control agency and at the EPOS side. This feedback primarily leads to a 'safe' situation as threatening or dangerous situations resulting from remote control can be dealt with directly, because operators on both sides of the WAN share a virtual space. Furthermore, this sharing of virtual space reduces the need for a physical relocation of personnel which saves travelling costs, resulting in a reduction of costs and the ease of setting up communication between project partners.

We have setup a collaborative engineering facility using low-cost hardware such as consumer market sound hardware, a webcam and a consumer market videograbber card. The software we used to create the virtual space consists of a set of multicast utilities known as mbone. These utilities are available as freeware. Configuration of these utilities is straightforward and an audio and video link can be achieved in a small amount of time.

Results

We found out that the sharing of a virtual space enhances the level of awareness among the operators, leading to a safe environment. Even if video frame rate is low, this gives enough information to collaborate effectively.

In this application it would be easy to use the public telephone network for the audio channel. However, this may not be the case for possible future applications. For example, if remote control is used over a space-ground link, then it is better to include audio in the same communications link as other information flows.

Future Developments

With the Remote Control project, we have laid the foundation for wide-area distributed simulations and operations. Future developments that may arise from the knowledge we have gathered include but are not limited to:

- *Whiteboarding:* Create the possibility for operators to draw figures and sketches on a

shared virtual whiteboard to quickly reach a common understanding.

- *Annotation:* Being able to write down texts or annotations on a common, digital notepad.
- *Integrating Command & Control workstations:* Adding applications such as the Advanced Crew Terminal to enable operations and training in a wide-area distributed simulation environment. Another point is that Origin has a long track record in the Command & Control (C^2) area of peer-to-peer collaborative working. For example, Origin's 3rd generation astronaut workstation, the Advanced Crew Terminal (ACT), features peer-to-peer collaboration between astronauts and ground-based scientists and controllers[6, 7].
- *Visualization:* Using our Virtual Environment (VE) it is possible to add a 3D visualization of the RVD within a fair amount of time.

Conclusions

It is shown that on many facets of simulation and the project life cycle Origin has followed a low-cost approach to facilitate the needs of clients and the market trends. Most, if not all of our attempts to reduce costs in the simulation market are based on the knowledge we gathered over the past years. We investigated the possibilities of low-cost simulation in a myriad of applications ranging from real-time distributed simulations with hardware-in-the-loop, man-in-the-loop wide area networking, to simulations with distributed intelligence, 3D visualizations, and command and control capabilities. The culmination of this knowledge is Origin's vision on a distributed simulation architecture based on a logical decomposition of components. As presented in this paper we recently added with success a wide-area network interface to the architecture. This allows a greater span of the distributed components and adds the cost-reducing feature of collaborative engineering without physical relocation to our low-cost approach.

As is shown, a wide area network interface can be added using low-cost ISDN channels. This connectivity, provided it is used in a point-to-point protocol is the basis for a reliable and controllable link between simulator or simulation components. Not only does it provide a facility to remote control simulation components, but it also realizes audio-visual links, which increases the level of awareness between remote operators.

With the successful results of the low-cost simulation research programme, Origin has the capability to meet the low-cost simulation demands not only in the space segment, but also in other markets.

References

1. Bremer, L. et al., 1998, *'Low-Cost Simulation: Architecture, Tools & Application'*, Proceedings of the 5th International Workshop on Simulation for European Space Programs (SESP), Noordwijk, The Netherlands.
2. Broeils, A. H. and Grant, T. J., 1999, *'A Constellation Planning Tool'*, International Astronautical Federation (IAF), The Hague, The Netherlands.
3. Broeils, A., Keijzer, J. & Grant, T., 1999, 'A Satellite Constellation Planning Tool: Application of a low-cost real-time simulation architecture', Proceedings of ESM '99.
4. Trompert, R. & Keijzer, J., 1999, *'Low-Cost Simulation: Architecture, Tools and Application to Rendezvous and Docking'*, Proceedings of DASIA '99, Lissabon, Portugal.
5. Schreutelkamp, E. and Heimbold, G., 1999 *'Low Cost Simulation with Hardware in the Loop'*, International Astronautical Federation (IAF), The Hague, The Netherlands.
6. Wolff, M. and Oijen, E. van, 1997, *'Advanced Crew Terminal: An Information and Control System for Microgravity Payload Operations'*, Preparing for the Future, vol. 7, no. 3, p. 18-20, ESA, Noordwijk, The Netherlands.
7. Oijen, E. van, Wolff, M. and Dujardin, P., 1999, *'The Advanced Crew Terminal, an Integrated, Common, Low-Cost and Configurable Toolset for Operations Activities'*, International Astronautical Federation (IAF), The Hague, The Netherlands.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37221

AIAA-2000-4084

DRAPER STATION SIMULATION - A STANDARD ARCHITECTURE FOR SPACE VEHICLE SIMULATION

Jiann-Woei Jang[1], Nazareth Bedrossian[2],
Jeremy Zoss[3], Jeremy Templeton[4], Edward McCants[5]
The Charles Stark Draper Laboratory, Inc.
2200 Space Park Dr, Suite 210
Houston, TX 77058

Abstract

Draper Station Simulation, a family of International Space Station on-orbit attitude control system simulations, has been developed and used for flight readiness certification. This architecture allows for both rapid turnaround and wide scope analysis. This high volume, high fidelity simulation features the following: (1) By standardizing the simulation architecture, it provides a platform to simulate any spacecraft attitude control systems. (2) With a well-defined plant database, it can be easily reconfigured for each Space Station assembly stage. (3) The self-contained data structure, which is used to document both simulation parameters and results, allows the user to track simulation results. (4) The compiled version increases the simulation speed significantly.

Introduction

In recent years, Draper Laboratory has been involved in supporting NASA Johnson Space Center activities related to assembly of the International Space Station (ISS) shown in Figure 1. Draper is a member on a NASA team that certifies the Space Station attitude control systems for flight readiness. To accomplish this goal, the Draper Space Station team developed a family of high fidelity simulations, Draper Station Simulation (DSS), to meet the demanding International Space Station schedule.

The evolving complexity of the overall system due to the large number of assembly stages as well as variety of control systems from various International Partners. Examples of controls systems include the Russian propulsive systems on the FGB cargo block, Service Module (SM) and Progress resupply vehicle, the US Control Moment Gyroscope (CMG) non-propulsive systems and the US propulsive options using the Interim Control Module (ICM) and Propulsive Module (PM), the Canadian Space Station Remote Manipulator System (SSRMS) as well as the Mobile Transporter (MT), Rotary Joint (RJ) controllers etc. Further, for new programs such as the Space Station it is desirable to have the flexibility to make rapid modeling changes as the system evolves and also to answer "what if" questions. This is required to stress test the system as well as evaluate its performance in the presence of uncertainties for robust performance verification. Another desirable feature for the simulation platform of choice is to require minimal resources for maintenance and upgrades. Overall, what is required is a rapid development and analysis capability with low overhead. For these reasons a Commercial Of The Shelf (COTS) based system was selected as the standard development and analysis platform.

In this paper, the Draper Station Simulation standardized software concept and architecture are presented. A specific example of the Russian Service Module propulsive motion control system simulation is reviewed. The common database standards are presented and the automated reconfiguration features are described. Execution speed improvement results for the compiled version of DSS as well as the verification results are also presented.

---

[1] Member Technical Staff II, The Charles Stark Draper Laboratory, Inc.; jjang@draper.com
[2] Principal Member Technical Staff, The Charles Stark Draper Laboratory, Inc.; naz@draper.com
[3] Engineer, Southwest Research Institute
[4] Graduate Student, Department of Mechanical Engineering, Stanford University
[5] Graduate Student, Department of Mechanical Engineering, Rice University
Copyright © 2000 by C. S. Draper Laboratory, Inc. Published by the American Institute of Aeronautics and Astronautics with Permission.

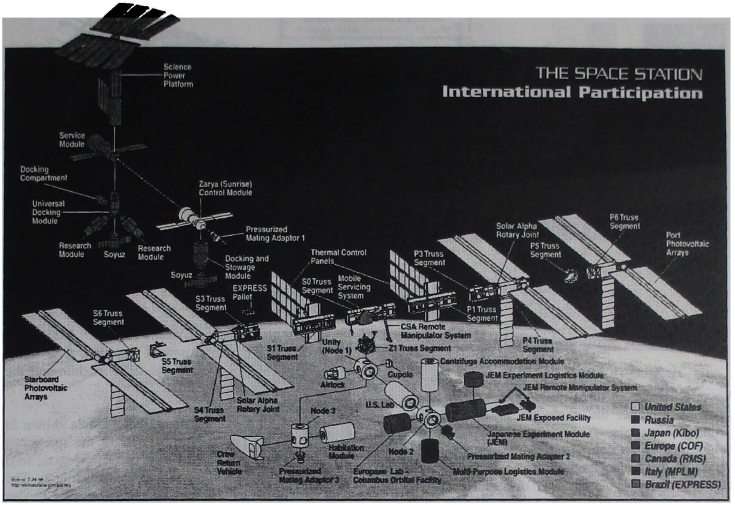


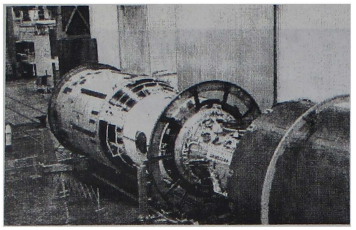
Figure 1. International Space Station Elements

Draper Station Simulation

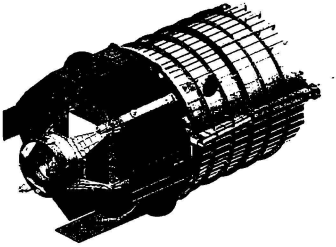
Draper Station Simulation is a family of International Space Station simulations using Matlab/Simulink [1] software as the COTS platform. This architecture allows "plug-and-play" modularity and rapid reconfiguration capability. DSS models the 3-Degree-Of-Freedom (DOF) attitude dynamics of a rigid/flexible space vehicle. It is used to evaluate potential for controller/flex structure dynamic interaction, control system robust stability and performance for different International Space Station operational modes, such as attitude hold, attitude maneuver, and reboost, and various Space Station assembly stages. It features a user friendly interface, quick setup, and reconfiguration and fast execution. It is also portable and can be run on a PC desktop, a PC laptop, or a Unix workstation.

The simulation architecture of DSS is designed to accommodate most attitude control systems. Currently, DSS supports two attitude control system models. The first is the Russian Service Module (DSS-SM) shown in Figure 2(a) while the second is the US Naval Research Laboratory Interim Control Module (DSS-ICM) shown in Figure 2(b). To simulate various International Space Station stages, DSS must be reconfigured to adequately describe the structural dynamics as well as the control system information for a particular ISS configuration. This reconfiguration process has been automated by using well-defined plant databases and model processing routines.

All the simulation parameters and the corresponding results are also stored in a self-contained database. The purpose of this database is to create a general structure, which is self-documenting and allows the user to track simulation results. DSS has been verified against NASA legacy simulations, and has become a validated NASA station flight certification tool.



(a). SM



(b). ICM

Figure 2. Service Module and Interim Control Module

Standard Simulation Architecture

The top-level architecture of DSS shown in Figure 3 consists of three main modules; the system dynamics module, the controller module, and the disturbance module. The system dynamics module includes system structural models in the state space form. The controller module contains the spacecraft attitude control system. The disturbance module models disturbance force/torque applied to the Space Station. This top-level simulation architecture covers most ISS operations, such as maneuver, hold, reboost, prox-ops and desaturation. The Russian Service Module motion control system [2] is used as an example to provide more detailed discussion of the simulation architecture.

In DSS, the system dynamics module consists of a multibody linear rigid/flex Space Station structural model in state space form. The structural model may also include appendage sub-systems such the Space Station Remote Manipulator System and also appendage controllers, such as solar array, thermal radiator, robotic joint etc. The controller module models the Service Module Motion Control System (SM MCS), and related sub-systems. The disturbance module is used to model external disturbance torques like gravity gradient and internal Euler coupling torques. The disturbance torques are utilized to emulate nonlinear structural dynamics via the use of d'Alembert's Principle [3].

In DSS-SM, the SM MCS model consists of four major modules: sensor, filter, attitude propagation, and Reaction Control System (RCS). The sensor module takes the environment rate first, then sends the measured rate to both the filter module and the attitude propagation module.

The purpose of the flex filter module shown in Figure 4 is to provide an estimate of the spacecraft rigid rates. It consists of four parts; low-pass, feed forward, disturbance and adaptive notch filters. The low-pass filter is used to attenuate bending modes and to minimize the effects of sensor noise and quantization. In estimating the rigid body vehicle rate, the flex filters rely on feedforward prediction of the jet firing rate change to account for the slow, .5Hz, sensor update rate and to allow use of a low bandwidth (~0.05Hz) low-pass filter. Additionally, a disturbance filter is also used to estimate slowly varying unmodeled disturbances. To attenuate low frequency bending modes that may not be adequately attenuated by the low-pass filter, a series of three second-order notch filters per axis are also used. The algorithms also provide for an adaptive tuning of the notch frequencies during flight to account for uncertainties in the structural model.

The attitude propagation module shown in Figure 5 integrates kinematic equations of motion [4] for one time step to generate the measured attitude quaternion. By subtracting the rate of the reference frame from the measured core-bode rate, the rate errors are calculated. By subtracting the command attitude from the measured attitude, the attitude errors are calculated. Both attitude and rate errors are then sent to the RCS module.

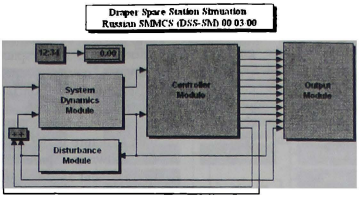


Figure 3. DSS Top-level Architecture

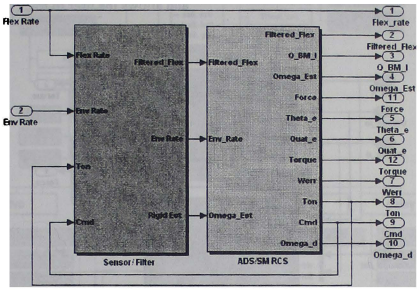


Figure 4. SM MCS

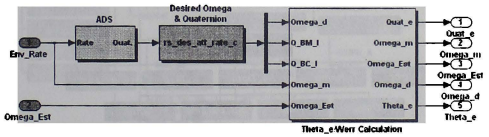


Figure 5. Attitude Propagation

In DSS-SM, the RCS module shown in Figure 6 consists of both Attitude Control System (ACS) and Thrust Vector Control (TVC) controllers. Both controllers take the attitude and rate errors, and then calculate the corresponding RCS forces and torques. The SM ACS controller, which consists of phase plane logic, maneuver logic, thruster delay logic, and thruster restriction logic, generates signed on-time commands and then the corresponding forces and torques.

The TVC controller is a simple Proportional-Derivative (PD) controller, which generates the gimbal commands. The TVC actuators adjust the main engine gimbal angles based on the gimbal commands. The corresponding forces and torques are calculated according to the current gimbal configuration. The combined forces and torques of the ACS and TVC together with the disturbance forces and torques are applied to the system dynamics module.

DSS Simulation Common Data Standards

The DSS takes full advantage of the Matlab environment with its use of objects and structures. Two main data structures are used in DSS-SM. The first is the plant database structure, *P*, shown in Figure 7. It contains the following ISS related plant information; state space model, input/output node names, mass properties, and jets thruster information/selection tables. With this well-defined database, DSS-SM simulations can easily be reconfigured for each ISS configuration. The second data structure is the closed-loop simulation database, *CL*, shown in Figure 8. This single database is used to document all the input/output information, such as plant dynamics, motion control system parameters, simulation parameters, and simulation outputs in a single structure. The purpose of this database is to create a general structure, which is self-documenting and allows the user to track simulation results.

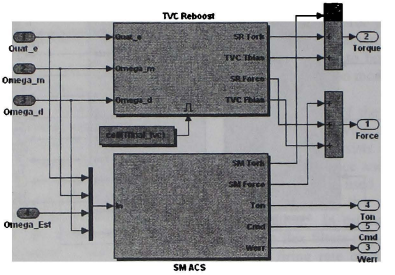


Figure 6. Service Module ACS and TVC

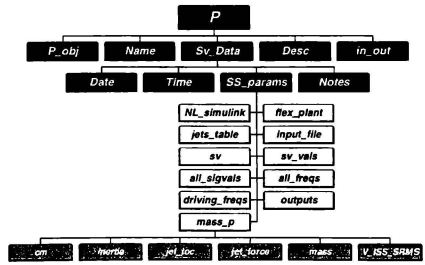


Figure 7. *P* database

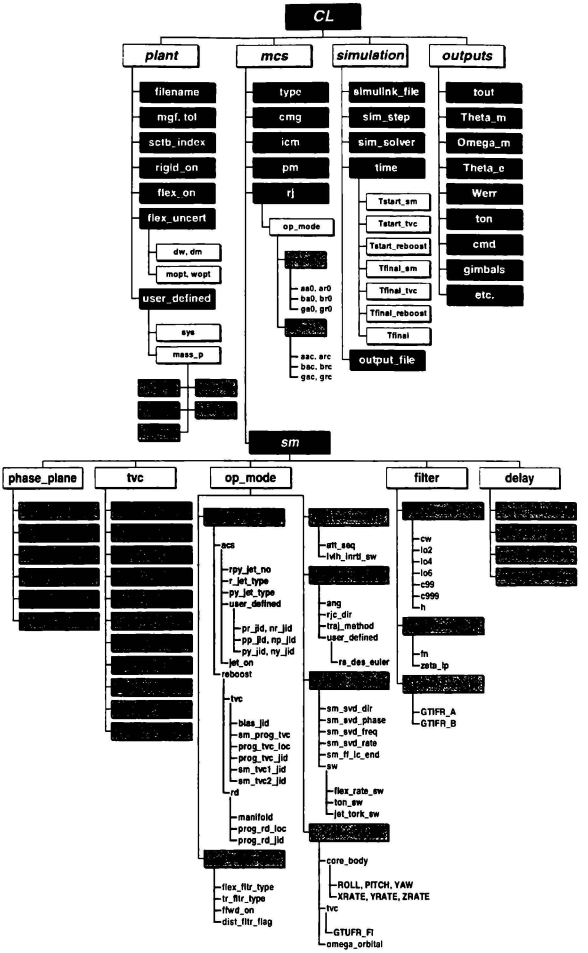


Figure 8. *CL* database

Automated Simulation Reconfiguration

Before proceeding to review the reconfiguration feature of DSS, a brief overview of ISS configuration changes during assembly is shown in Figure 9 to Figure 12. Each of these figures shows a particular stage in the Station assembly process and the corresponding ISS structure, motion control system, ACS engine/thruster as well as robotics information. Figure 9 shows the characteristics for Stage 1R, which is currently the next stage in the assembly sequence. Stage 1R is formed when the Russian Service Module vehicle docks to the FGB module already in orbit. Figure 10 depicts Stage 4A during robotic assembly using the Space Shuttle Remote Manipulator System (SRMS) with payload the first US solar arrays. Figure 11 shows the completed assembly for Stage 4A, when the Beta RJ controllers are first appeared, and Figure 12 provides detail on Stage 13A.

Simulation reconfiguration in DSS for each ISS assembly stage is accomplished via an automated rewiring process which utilizes the well-defined *P* database. The automated re-wiring process is shown in Figure 13. To implement this approach, first, a generic DSS model is saved in the simulation database. Next, the ISS plant state space models are stored in the *P* database with all the input names and output names clearly defined and recorded. Finally, all the ISS non-propulsive controller models, namely Alpha RJ, Beta RJ, Gamma RJ, and Controller Moment Gyro (CMG) controllers, are stored in the controller database. To re-configure each ISS stage, first the DSS template is copied from the simulation database. Next, the ISS dynamics block is removed and replaced with the current dynamics module. The current dynamics model is constructed based on the input names and output names stored in the *P* database. For example, the ISS Stage 12A+OBS model has four pairs of Beta Resolvers/Tachometers and one pair of Alpha Resolver/Tachometer outputs shown in the output name list, and has four Beta Torque and one Alpha Torque inputs shown in the input list. Once the rewiring algorithm detects the existence of matches between the input name list and the output name list, automatically copies the corresponding model from the controller database and connects the controller with the plant at proper input and output node locations. When the plant and controllers are properly connected, the resulting current dynamics block is then attached to DSS to complete the reconfiguration process.

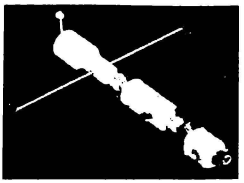
Simulation Speed Enhancement Via Compiled Code

The prototype DSS was first developed using Matlab pseudocode which is user friendly and allows rapid simulation development with minimal programming overhead. However, due to its interpreter nature, it is not suitable for large number of simulations especially when a simulation has reached a mature stage and few changes are expected or required. In order to enhance execution speed, all the pseudocode functions were compiled using the SIMULINK C MEX-file S-function format.

To illustrate the improvement in sim execution speed, three testcase ISS models, namely Stage 4A, 5A, and 4A+OBS+USLAB were used. The visualizations of non-zero elements in the structural dynamics matrix for the testcase models are shown in Figure 14. As it will be shown later, the sparseness of the dynamics matrix will affect the simulation speed significantly. The more non-zero elements the matrix has, the more run-time it will take for the same simulation interval. All the numerical examples shown in this paper were generated from a SUN Ultra-2 workstation. All versions of DSS were implemented in Matlab/Simulink version 5.2.1/2.2.1 and Real Time Workshop (RTW) version 5.2.1. The RTW version of DSS is a stand-alone ISS simulation package, which uses RTW to compile the C-MEX version DSS.

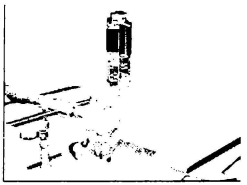
The speed comparison among four different versions of DSS [5], namely Matlab pseudocode (prototype), M-file S-function version, C MEX-file S-function version, and RTW version, are shown in Figure 15(a). The conversion from Matlab pseudocode to M-file S-function improves the simulation speed by 50%. However, significant improvement in simulation execution speed can be gained by translating all M-file S-functions to compiled C MEX-file S-functions. The simulation speed enhancement for the C-MEX file S-function version when compared to Matlab pseudocode increased by 2609%, 1632% and 677% for 4A, 5A, and 4A+OBS+USLAB modes.

The numerical comparison between C MEX-file S-function version and RTW version are shown in Figure 15(b). In general, the RTW version shows better performance for sparse dyamics model simulations, but is ineffective for computationally intensive simulations, such as 4A+OBS+USLAB testcase.



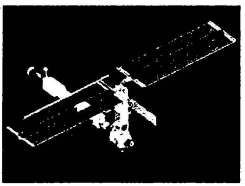
Structure	ISS 1R
Output No	6
Input No	16
Modes	141
ACS Controller	SM
ACS Thrusters	SM
Reboost Engine	-
RJ Controllers	-
Robotics	-

Figure 9. Stage 1R: Configuration



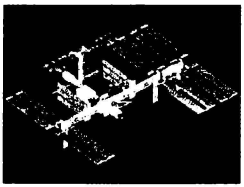
Structure	ISS 4A + Orbiter
Output No	30
Input No	16
Modes	98
ACS Controller	SM
ACS Thrusters	SM
Reboost Engine	Progress, SM
RJ Controllers	-
Robotics	SRMS

Figure 10. Stage 4A+Orbiter+Robotics: Configuration



Structure	ISS 4A
Output No	10
Input No	61
Modes	375
ACS Controller	SM
ACS Thruster	SM
Reboost Engine	Progress, SM
RJ Controller	Beta(2)
Robotics	-

Figure 11. Stage 4A: Configuration



Structure	ISS 13A
Output No	42
Input No	41
Modes	838
ACS Controller	SM, CMG
ACS Thruster	SM, SPP, Progress
Reboost Engine	Progress, SM
RJ Controller	Alpha(4), Beta(6), Gamma (2)
Robotics	SSRMS, MT

Figure 12. Stage 13A: Configuration

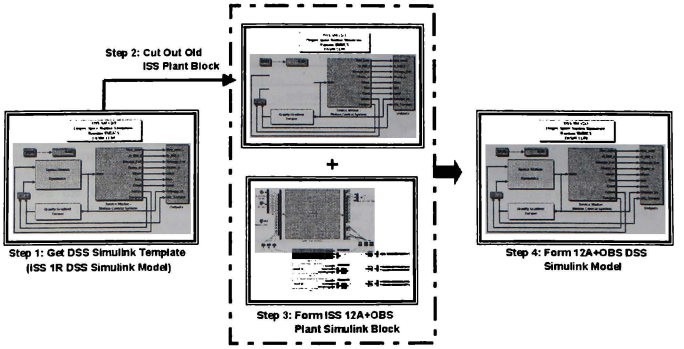
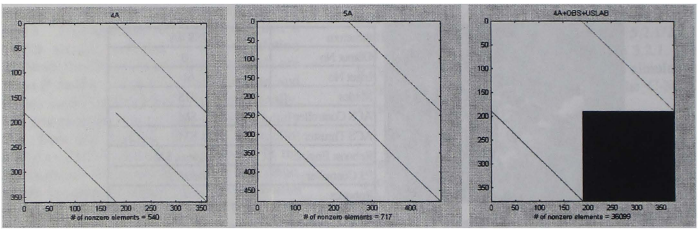


Figure 13. Automatically Reconfiguration Mechanism

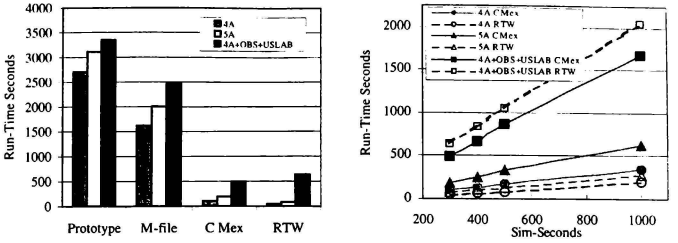


4A 5A 4A+OBS+USLAB

Figure 14. Non-zero Element in ISS State Space Model

Simulation Verification

To verify DSS-SM simulation fidelity, several rigid body simulation results from DSS-SM are compared to results from the NASA certified Space Station Multi-Rigid Body Simulation (SSMRBS) [6]. SSMRBS is a non-COTS based legacy, high-fidelity 6-DOF C/Fortran based simulation which includes the SM MCS. The verification example details are presented in [6]. The illustrative DSS-SM/SSMRBS comparison results are shown in Figure 16. Figure 16(a)-(d) compares measured rate, attitude error, gravity gradient torque, and propellant usage. The verification results indicate a near exact match with SSMRBS. More extensive verification examples are presented in [7]. It is concluded that DSS-SM exhibits near identical behavior when compared to SSMRBS.



(a) 300 Seconds Simulation

(b) C Mex Version Vs. RTW version

Figure 15. Comparison of DSS-SM MCS Speed Enhancements

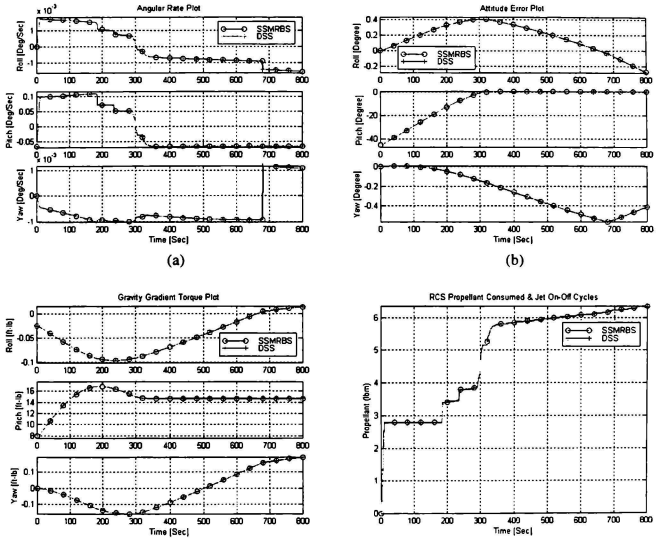


Figure 16. Simulation Comparison between DSS and SSMRBS

Conclusions

In this paper, the Draper Station Simulation (DSS) standardized software concept and architecture was presented. A specific example of the Russian Service Module motion control system simulation was reviewed. The common database standards were outlined and the automated reconfiguration features were described. Execution speed improvement results for the compiled version of DSS were presented, as well as verification against a NASA certified high-fidelity simulation. It is concluded that a rapid development COTS based simulation analysis platform coupled with a judicious choice for software concept and architecture can be a very effective and efficient tool for flight certification. It also provides desirable capabilities complementary to legacy based simulations. Extensive experience with DSS-SM over the previous three years has shown the benefits of such an approach.

References

[1]. SIMULINK - Dynamics System Simulation for MATLAB: Using SIMULINK, The Mathworks Inc., Natick, Massachusetts, January 1997.

[2]. Jackson, M., Lacovara, R., Zimpfer, D. and LoPresti, J., "Integration and Users Guide of the Russian Segment of the International Space Station Attitude Control System," Draper Memo SSDI-99-010, Nov., 1999.

[3]. Clough, R. W., Penzien, J., Dynamics of Structures, McGraw-Hill, Inc., New York, 1975.

[4]. Wertz, J.R., ed., Spacecraft Attitude Determination and Control, D. Reidel Publishers Company, Holland, 1980.

[5]. Templeton, J., Zoss, J., "Compiled MATLAB/ SIMULINK Simulations: DSS Experience," Draper Memo No. SSDI-98-005, Draper File No. E41-98-072, July 8, 1998.

[6]. Jackson, M., "Validation Data for Service Module RCS Model," JSC/AFMD/EG4, July. 7, 1999.

[7]. Jang. J., "Draper Station Simulation-Service Module: Version 4.1 Validation," Draper Memo No. SSDI-00-012, Draper File No. GCC2-00-044, March 14, 2000.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37222

AIAA 2000-4085

Simulators for the X-38/CRV parafoil and re-entry phases

Frank J. Delgado [a], Marco Arcioni [b]

[a] Intelligent Systems Branch/ER2
NASA/Johnson Space Center,
Houston, TX 77058
fdelgado@ems.jsc.nasa.gov

[b] Modelling and Simulation Section/TOS-EMM
ESA/European Space Technology Centre
Noordwijk, The Netherlands 2200AG
marcioni@estec.esa.nl

ABSTRACT

The NASA Johnson Space Center, in conjunction with the European Space Agency is developing a series of prototype flight test vehicles that will lead to a functional Crew Return Vehicle (CRV). The operational CRV will be attached to the International Space Station and serve as a Space Station lifeboat to ensure the safe return of crew members to Earth in case of an emergency. The development of these prototype vehicles, designated as the X-38 program, will demonstrate which technologies are needed to build an inexpensive, safe, and reliable spacecraft that can rapidly return astronauts from onboard the International Space Station (ISS) to Earth. The X38/CRV is unique in several aspects: it does not afford the pilot a forward view through a window and it utilizes a parafoil in the final landing flight phase. As a result of some of these unique attributes the prototype on-board displays will need to provide enhanced situation awareness to aid a crew member in supervising the onboard GNC software to select a landing site, and possible obstacle avoidance during the final flight phase. In the ESA/ESTEC visualization prototype system the scene is visualized using MultiGen OpenFlight models on high-end SGI machines. The Parafoil Guidance Navigation and Control (PGNC) algorithms are being developed as a joint NASA/ESA venture. These algorithms will be used onboard the vehicle to create nominal flight trajectories for the vehicle to follow. These same algorithms are also used as parafoil simulators in our augmented situation awareness system we are developing. The output of the parafoil simulator is displayed on a Windows based computer running the LandForm FlightVision software. The FlightVision software is used to create the synthetic environment displays and all the necessary HUD symbology. Maps, such as aeronautical charts, as well as satellite imagery are optionally overlaid on the 3D-terrain model to provide additional situation awareness for the crew.

Keywords: Synthetic vision, simulation, flight visualization, flight guidance, human factors.

INTRODUCTION

The X-38 program began in early 1995 to explore the feasibility of building a space station Crew Return Vehicle (CRV). The X-38 program is developing a series of test vehicles to demonstrate the low-cost technologies and methods required to develop a fully functional CRV that can rapidly return astronauts from onboard the International Space Station (ISS) to Earth. The X-38 program uses a gradual buildup approach. Three atmospheric test vehicles and one space rated vehicle will be developed and tested during the X-38 program. The atmospheric test vehicles are known as vehicle 131 (V131), vehicle 132 (V132), and vehicle 131R (V131R). The space-rated vehicle that will fly on

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental Purposes. All other rights are reserved by the copyright owner

the Shuttle as a payload bay experiment in April 2002 is known as vehicle 201 (V201).

The X-38 employs a "lifting body" concept originally developed by the U.S. Air Force's X-24A project in the mid-1960s. The concept uses the aerodynamic shape of the vehicle itself to generate the lift that a normal aircraft gets from its wings. This gives the X-38 vehicle good re-entry maneuverability capabilities. More important, as a lifting body, the X-38 has excellent cross-range characteristics. These cross-range characteristics assure multiple opportunities for a dry terrain landing within the 9-hour lifetime of the vehicle consumables. The ability to return to Earth quickly is very important and is a major advantage that the X-38 CRV has over the Russian Soyuz capsule, which is also under consideration for possible use as a CRV. Unfortunately, the Soyuz has two major drawbacks: its inability to accommodate crewmembers that vary greatly in size and its inability to carry more than three crew members at a time. These issues caused concerns because when fully operational the ISS will house up to 7 crewmembers that can vary greatly in size (5th percentile Asian female to 95th percentile U.S. male). Because of these concerns, an investigation into the development of an alternate method of returning crewmembers to Earth was launched. This effort is known as the X-38 program. The X-38/CRV is being designed to accommodate the necessary range of crewmember sizes and have the capability of carrying 7 crewmembers at any time.

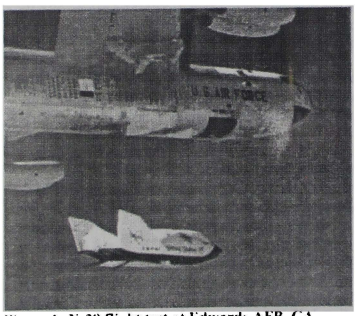


Figure 1: X-38 flight test at Edwards AFB, CA.

V131, V132, and V131R have composite fiberglass bodies and have/will undergo extensive testing during captive carry tests and free flight tests. During a captive carry test, a vehicle is attached to the wing of a B-52 and flown at different velocities and altitudes to collect data. During a free flight test the vehicle is carried to an altitude between 20k and 50k feet, under the wing of the B-52 and then released (Figure 1). The vehicle flies "free" for several seconds before a large parafoil is deployed and used to return the vehicle safely to the ground.

V131 has undergone 7 captive carry tests and 3 free flight tests. V132 has undergone 5 captive carry tests and 3 free flight tests. V131R and V133 are atmospheric test vehicles similar to V132 and are currently being built at JSC. V201 will be the first space-rated vehicle built in-house at the Johnson Space Center. It will be taken into space on the Space Shuttle April, 2002. Once in space, V201 will be taken out of Shuttle payload bay by the Remote Manipulator System and released. V201 will then run through its automated check out procedures and begin it's de-orbit sequence. The vehicle will then enter Earth's atmosphere and at an altitude of approximately 30k feet begin the deployment sequence for the large steerable parafoil. The large parafoil has active guidance from an on-board GPS receiver and will be used to navigate the spacecraft safely back to the ground

The current X-38 CRV mission requirements include returning up to 7 crewmembers from the ISS safely to Earth, have the ability to insure a dry terrain landing, and have enough cross-range to insure three landing opportunities in nine hours. This would be done in the event that any of the following situations arise: an ISS catastrophe, an emergency medical evacuation, or if the Space Shuttle is unavailable to re-supply the ISS. Because we must design to a worst case scenario, a medical emergency where crewmembers are unable to pilot the vehicle back to Earth, a fully autonomous vehicle must be built.

The basic assumption that a pilot is not necessary to return the CRV to Earth meant that a forward-looking window is not a hard-set requirement for the CRV. Although full autonomy is necessary for the medical evacuation scenario, keeping crewmembers in the loop to take care of unforeseen situations whenever possible is also a must. Synthetic environment technology is ideal for augmenting a crewmembers situational awareness and helping them to reselect a landing site or to manually fly the vehicle during the parafoil flight phase. As a result of some of these unique attributes the prototype on-board displays will need to provide enhanced situation awareness to aid a crew member in supervising the onboard GNC software to select a landing site, and to provide obstacle avoidance information during the final flight phase.

The augmented situational awareness system being developed is composed of two major components: a

display component that uses state-of-the-art augmented reality technology and several robust/dynamic simulators. NASA and ESA originally developed the simulators we are currently using to develop our synthetic environment system. The system developed has been used to:

1) Monitor live X-38 captive carry and free flight testing. In this role it acts as a tele-presence tool.

2) Analyze and playback flight data.

3) Evaluate different levels of crew interaction. By combining the simulators with the displays developed, we are capable of "manually flying" the parafoil flight phase of a simulated X-38 flight mission. This system is helping define what level of crew interaction will be allowed in the final Crew Return Vehicle.

The X-38 uses a large parafoil during the final flight phase. This parafoil is the subject of a significant engineering effort and considerable effort has been spent testing the parafoil system. Testing is done using large instrumented pallets attached to the parafoil. The pallets are released from the back of a C-130 aircraft and data on the aerodynamics, deployment sequence, and overall performance are recorded and closely analyzed. Early testing has yielded an ample amount of data, which has been used to successfully build a parafoil system that has safely returned V131 and V132 back to the ground during free flight tests. This data has been fed to our synthetic environment displays. Combining augmented reality technology to tele-operation can considerably reduce the risks of remotely operating aircraft.

The approach selected to perform our task is an incremental one. We began our display development by creating military compliant HUD symbology for our displays. Since astronauts would eventually use the displays, we also added symbology that conformed to Shuttle HUD symbology standards. After the initial symbology development, we allowed the displays to be evaluated by several astronauts, engineers, and operations personnel. These individuals provided feedback on what symbology was more applicable for our project and suggested additional symbology to develop. We incorporated their suggestions into our displays and began an iterative process where we would collect feedback, implement, show, collect feedback, implement, show, etc. This iterative process has allowed for the definition of information rich displays that have been successfully deployed and tested in several key areas of the X-38 program.

As astronauts came in contact with the system being developed, it became apparent that it could also serve as a crew-training tool that would considerably improve a pilot's situational awareness as a ground-based or onboard display.



Dryden Flight Research Center EC99-44888-44 Photographed 06FEB1999
NASA's X-38, a prototype of a Crew Return Vehicle (CRV) completes its second free flight. NASA Dryden/Carla Thomas

Figure 2: Parafoil deployed prior to landing.

For the system developed to be used in particular situations, the system would have to be integrated into other NASA systems. This has required that the software being used to model and render the virtual scenes to be turned into embeddable software components. Rapid Imaging Software (RIS) has delivered an embeddable version of the core routines necessary to integrate our software with other NASA developed systems/software.

Re-entry Flight Phase Visualization

Because of the incremental approach selected by the project, most of the resources have been concentrated on the atmospheric flight phase of the X-38 vehicle. Due to this fact, the re-entry flight phase visualization symbology is not as well defined as the parafoil flight phase symbology. The development of a system that can be used to quickly and accurately visualize the re-entry flight phase can become an invaluable tool to fine-tune the GNC algorithms and provide astronauts, engineers, and operations personnel with a powerful

training and diagnostics tool. In the frame of laboratory research activities, ESA/ESTEC has developed a re-entry flight phase simulation and visualization prototype that could be applied to the X38/V201 re-entry phase. this is based on the EuroSim real-time simulator shell. The ALTOS software is used to generate a re-entry trajectory that is based on a landing site, a re-entry point, and additional constraints; such as, the maximum heating flux and the g load supported by the X38 vehicle. The simulated re-entry trajectory, calculated using a Matlab/Simulink real-time GNC that has been converted into a EuroSim model, generates trajectory data that is displayed in real-time through a 3-D synthetic environment visualization application. This visualization is performed on a high-end Silicon Graphics workstation running the commercially available VEGA software from MultiGen-Paradigm. State of the art modelling and prototyping techniques and automatic levels of details in the models are achieved using MultiGen Creator II. The special effects in the scene; such as, a tiled Earth model and the billboard used to generate the atmosphere around the Earth, are created by using the Performer libraries directly. A simplistic block diagram of the simulator and the visualization system is depicted in Figure 3.

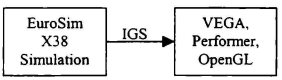


Figure 3: Block diagram of the X38 re-entry phase simulation/visualization system.

The EuroSim engine generates real-time position and attitude information for all the objects present in the scene. Some of the objects present include: the Earth, Moon, Sun, stars, the Space Station with its rotating solar panels, an X38 vehicle, and an assortment of instruments on a simulated instrumentation panel. The behavior of the objects is based on internal models written in C and Fortran. VEGA is used to drive the MultiGen OpenFlight models and takes care of the real-time rendering of the scene. The communication between the two main blocks is done using an IGS interface, this takes care of passing the 6 degrees of freedom data for each moving part, from EuroSim to VEGA. The IGS interface also permits for the definition of multiple viewpoints around the Earth with the ability to pan and zoom a virtual camera positioned either inside or outside the X38 vehicle. The zoom and pan feature can also be performed to include a view of the International Space Station (ISS).

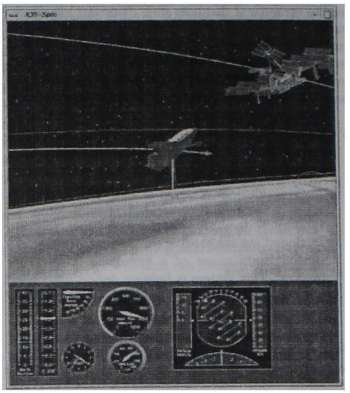


Figure 4: Observe the difference between the nominal re-entry path and the X38 simulated trail.

The virtual scene depicted in Figure 4 contains a nominal re-entry trajectory and a simulated X38 trajectory. The visualization system developed at ESTEC for the X38 vehicle includes both out-the-window 3D visuals and cockpit instrumentation visuals. In the Figures 4 and 5 we see a typical re-entry display that includes a 3D scene on the top and an instrumentation panel on the bottom. Objects such as dials, tapes and pitch ladders can be created and maintained easily with MultiGen Creator II. These objects can be plugged in the simulated cockpit panel and deployed for visualization in the same display window as the out-the-window visuals. Although, the system developed provides excellent situational awareness, the system that will be deployed for the final CRV is still being defined. To better define the symbology and displays required during the re-entry flight phase, close interaction with the several astronauts and engineers have recently begun. A preliminary set of displays that includes the astronaut/engineers feedback will be developed during the next 3 months.

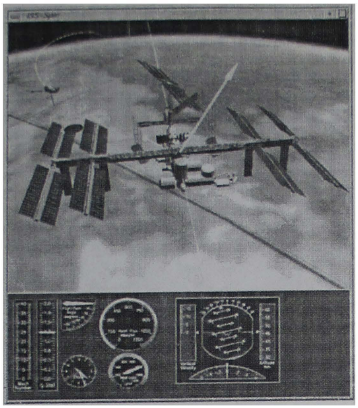


Figure 5: X38 Re-entry phase imagery with a main view of the ISS.

Parafoil phase visualization

The visualization of the parafoil flight phase will be performed with the use of the Landform FlightVision modeler. FlightVision is a commercial off the shelf software product sold by Rapid Imaging Software. Their products can be purchased as stand-alone software or embeddable software components. We have developed our system using the stand-alone FlightVision software, but are currently migrating it to use the embeddable FlightVision software components. Fundamentally our system must provide a real-time three-dimensional display of the environment, incorporating diverse terrain, navigation, and aircraft data. This display must be a natural perspective display from a viewpoint that is controlled by six degrees of freedom (6 DOF) data. The 6 DOF data is obtained by one of the following three methods:

1) From an embedded Global Position System / Inertial Navigation System (GPS/INS) that resides onboard the X38 test vehicles and telemeters it's data to the ground stations during live flight testing.
2) From a simulator that resides on one of the simulation machines.
3) From pre-recorded trajectory flight files.

The camera modeled by the software must be a perspective camera that can be placed at any point in our 3-D virtual environment, and rotated about 3 axes to any orientation to simulate any viewpoint in which a real camera might be found. This 3-D synthetic vision of the world must also incorporate diverse elements including:

- Land surface topography;
- Textures (satellite and aerial imagery);
- Ground-fixed objects like landing zones, buildings and obstacles in general;
- Head-up displays for important information display concerning the actual flight situation.

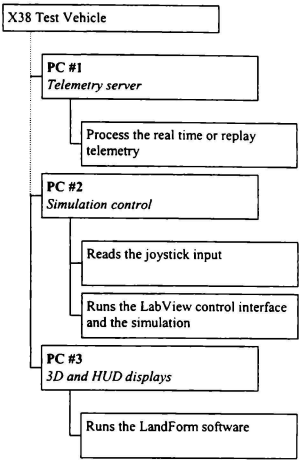


Figure 6: Block diagram of the X38 parafoil phase visualization/simulation system.

The schematic diagram for the synthetic environment parafoil visualization system is depicted in Figure 6. It consists of three programs that are currently running on separate computers.

The first computer is the ground server machine that processes the X38 vehicle data and distributes it to ground system display machines, and to the second computer (the simulation computer).

The second computer, also called the simulation computer, is responsibility for providing the third computer, the visualization computer, with the necessary data. The data distributed by the second computer can be from an embedded parafoil simulator or from the server computer. There are currently two parafoil simulator algorithms embedded in this computer: the Parafoil GNC (PGNC) algorithms, and the Parafoil Dynamic Simulator (PDS). Both of these algorithms are housed in a stand-alone Labview program. The user-entered commands in the Labview interface allow the operator to switch between different modes of operation with ease. Different camera views can be selected (internal view along the velocity vector, fixed camera view, chase view and map views (both North-up or forward-up). Three different simulator modes of operation can be selected: a fully automatic PGNC mode, a fully manual mode (in this case the joystick and keyboard inputs are accepted), and a semi-auto mode (where the control loop is open, the winches control is manual but the PGNC is on and gives commands suggestions based on its nominal landing path). The camera view parameters, mode parameters, and all the necessary flight data parameters, either from X38 vehicle or from the simulator, are passed to the third computer.

The third computer, also called the visualization computer, hosts the Landform FlightVision software. The FlightVision software, developed by Rapid Imaging Software, is used to render the virtual scene and all its associated symbology. This includes rendering the 2D view, the 3D view that includes elevation terrain models, ideal and actual glide trajectories, recommended landing areas, and typical HUD symbology. Maps, such as aeronautical charts, as well as satellite imagery are optionally overlaid on the 3D-terrain model to provide additional situation awareness. The FlightVision software can be augmented with dynamic linked libraries (dll's) that users can create. The dll's allow one to develop custom HUD symbology, entity models, and flight tracks that can be incorporated into the FlightVision generated scene. Figure 7.0 and Figure 8.0 are typical HUD enabled displays that have been defined with the aid of NASA/ESA astronauts. The 2D and 3D displays are dynamically up-dated and are currently used to display real-time flight data coming from the ground system (e.g. during a test flight) or from one of the parafoil simulators.

The system developed is currently deployed in the X38 Mission control center, in the Remote Cockpit Van (RCV), and at desk based simulation stations. The RCV is unique because it is a mobile testbed that contain all the information necessary to perform flight following activities during live X38 flight-testing. The first attempt to remotely pilot the parafoil flight phase of V131R from the RCV will be performed later this year.

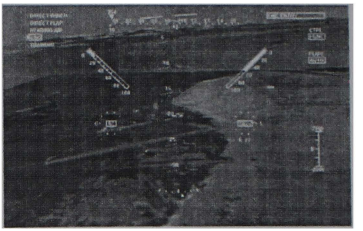


Figure 7: This is an immersive 3D scene generated with the Landform FlightVision software. In this scene we can see the elevation terrain models, the imagery draped on the terrain, a pitch ladder, a bank indicator, a heading indicator, a winches position indicator, an altitude indicator, and an airspeed/groundspeed indicator.

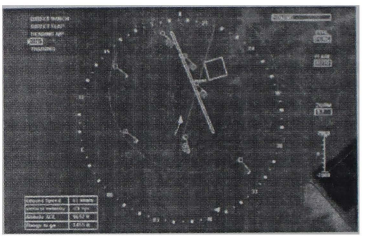


Figure 8: This is a 2D scene generated with the Landform's FlightVision software. In this scene we can see a custom HUD that is superimposed on an image. Some of the custom HUD symbology depicted includes: a wind-rose heading indicator, wind direction indicators spaced along a depicted nominal trajectory, a runway model, text boxes to indicate the flight phase, and an altitude indicator.

Conclusion

The ESTEC X38 re-entry test bed is a very good ground/tele-presence visualization system because of the realism provided in the rendered scene. It can be used as a powerful tool for displaying prototype displays, and as an analysis tool to fine-tune the on orbit trajectory algorithms. The general definition of

such system can permit one to extend the system from a re-entry only system to include other flight phases of the X38/CRV. An end-to-end system could be developed with this approach if the system incorporated algorithms that simulated all the different flight phases of the X38. The NASA/JSC work has demonstrated that a real-time synthetic vision system can be developed to augment the situational awareness of a remotely piloted aircraft/space craft. This same type of system can also be used by astronauts onboard an X38/CRV type vehicle to augment their situational awareness.

Current plans are to continue this effort and demonstrate a hybrid synthetic vision system that combines real-time video with FlightVision generated displays, this is known as the 3-D HUD concept. We will also examine the applicability of this technology to Helmet-Mounted Displays, and other displays mediums that may be appropriate for use in an X38 type vehicle. Current efforts are focused on the development of a system that will support V131R, which will begin flight July 2000, and V201, which is scheduled to go up on the Space Shuttle April 2002.

Acknowledgements

We would like to thank the following astronauts: Scott Altman, Ken Ham, Alan Poindexter, Frank De Winne, Rick Husband, and Philippe Perrin for their helpful suggestions in the HUD definition. A thank goes to Mike Abernathy of Rapid Imaging Software, Inc. for his technical support on the LandForm product line and his insights in the development of HUD symbology using OpenGL. A thank also to DASA's engineers U. Soppa and H. Strauch for their help on the PGNC interfacing. A thank goes to Nicolas Graf (NASA/JSC), Antoine Dumurgier (SUPAERO), Nicolas Pen (European Space Agency), Fabrice Cinquetti (European Space Agency), and Mathieu Peyrega (SUPAERO) for their support during the coding of the initial HUD symbology. Thanks also to Simon Mills and Perter Van Der Plas of ESA/ESTEC TOS/EMM for their support on VEGA and Eurosim.

References

1. J. F. Muratore and C. S. Iacomini, "Parafoil Flight Test of X-38 Prototype Crew Return Vehicle Yields Improved Instrumentation and Techniques", *Journal of International Flight Test and Evaluation Association*, pp 19-24, June/July 1998, Volume 19, Number 2.
2. F. J. Delgado, et al, "Real-Time 3-D Flight Guidance with Terrain for the X-38", *Enhanced and Synthetic Vision 1999.*
3. U. Soppa, et al., "GNC concept for the X38 Parafoil Descent and Landing Phase", X38 Flight Dynamics Team, NASA/JSC Note#216, January 1999.
4. U. Soppa, et al., " Parafoil GNC Ground Station Man Machine Interface", ", X38 Flight Dynamics Team, NASA/JSC, Spetember 1999.
5. U. Termote et al., "EuroSim RT Engineering Simulation Tool 2000 and Beyond: Automated Transfer Vehicle Application", to be presented at the AIAA Modeling and Simulation Technology Conference, Denver, August 2000.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37223 AIAA-2000-4086

Atmospheric Re-entry Modeling and Simulation: Application to the Crew Return Vehicle

R. R. Costa,* J. A. Silva,† S. F. Wu,‡ Q. P. Chu,§ and J. A. Mulder¶
Faculty of Aerospace Engineering, TUDelft, Delft, The Netherlands

The atmospheric re-entry modeling and simulation has been studied in the past year at the Control and Simulation Division of the Faculty of Aerospace Engineering. This study, done on the behalf of the European Space Agency, was focused on the Crew Return Vehicle (CRV) vehicle. Within the scope of this research, a simulation tool of the atmospheric re-entry was implemented. The simulation tool consists of three parts: simulation of the re-entry dynamics, simulation of the vehicle characteristics, and simulation of the environment. The modeling of the re-entry dynamics was performed in order to obtain a general tool for re-entry simulation (GESARED). The modeling of the CRV consisted in the implementation of the aerodynamic database cycle 8 (Rev. 8.3) and the modeling of the sensors and actuators of the vehicle. Atmospheric models (GRAM-99 and Standard Atmosphere 1962) and gravitation models were implemented to simulate the environment. The correctness and validity of the simulation tool was verified by numerical tests. The ARD (Atmospheric Re-entry Demonstrator) vehicle is now being implemented in GESARED, to perform tests on its re-entry in Earth's atmosphere. Future work on simulation of a Mars entry is planned.

Nomenclature

δ	Latitude
μ	Earth's gravitational constant
$\mu_{S/H}$	Bridging function Subsonic/Supersonic
μ_{RAR}	Bridging function rarefaction
Ω_E	Planet angular velocity
$\bar{\omega}$	Angular velocity
ϕ	Co-latitude
ρ	Density
ρ_i	Density in the beginning of the i_{th} layer
τ	Longitude
$\bar{\theta}$	Angular position
θ	Angle of incidence
C_x	Generic aerodynamic coefficient
C_{x_S}	Subsonic/Supersonic aerodynamic coefficient
C_{x_H}	Hypersonic generic aerodynamic coefficient
$\bar{F}_A$	Aerodynamic force vector
$\bar{F}_E$	External force vector
$\bar{F}_g$	Gravity force vector
$\bar{F}_T$	Thruster force vector
$\bar{g}$	Gravity acceleration vector
g_0	Gravity acceleration at sea level
$\bar{H}$	angular momentum vector
h	Altitude
I	Inertia matrix
J_i	Jeffery constant of ith order
L_{Z_i}	Thermal lapse rate
$\bar{M}_A$	Aerodynamic moment vector
$\bar{M}_E$	External momentum vector
$\bar{M}_T$	Thruster moment vector
m	Mass of the vehicle
P	Pressure
P_0	Total pressure
P_i	Pressure in the beginning of the i_{th} layer
P_L	Measured pressure
p,q,r	Angular velocities in Body frame
Q_i	i^{th} Quaternion
q_c	Dynamic pressure
R	Distance towards the center of Earth
R_E	Radius of Earth
R_e	Equatorial radius of Earth
R_g	Gas constant for air
R_p	Polar radius of Earth
$\bar{r}_{cm}$	Position of the center of mass
T_M	Molecular Temperature
T_{M_i}	Molecular Temperature in the ith layer
$\bar{V}$	Velocity vector
v_δ	Velocity in North direction
v_r	Velocity in East direction
v_R	Velocity in down direction
Z	Geometric altitude

*Graduate Student, Control and Simulation Division, Faculty of Aerospace Engineering, E-mail:R.R.Costa@LR.TUDelft.NL

†Graduate Student, Control and Simulation Division, Faculty of Aerospace Engineering, E-mail:J.G.AraujodaSilva@student.tudelft.nl

‡Research Fellow, Control and Simulation Division, Faculty of Aerospace Engineering, E-mail:S.F.Wu@LR.TUDelft.NL

§Assistant Professor, Control and Simulation Division, Faculty of Aerospace Engineering, E-mail:Q.P.Chu@LR.TUDelft.NL

¶Professor, Chairman of the Control and Simulation Division, Faculty of Aerospace Engineering, E-mail:J.A.Mulder@LR.TUDelft.NL

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

Z_i Geometric altitude in the ith layer

Introduction

ATMOSPHERIC re-entry is nowadays one of the main research fields in space technology. In order to provide an environment to design control laws for re-entry vehicles, TUDelft is developing a simulation tool of atmospheric re-entry. This simulation tool is called GESARED (GEneral Simulator for Atmospheric Re-Entry Dynamics), and was implemented in MATLAB/SIMULINK.

GESARED was initially developed under the contract "Fuzzy Logic for Spacecraft Control". This contract was worked out by the Control and Simulation Division of the Faculty of Aerospace Engineering on the behalf of ESA.[1] In this contract, GESARED was made with the purpose of simulating the re-entry phase of the Crew Return Vehicle. GESARED was the open loop plant, and the loop was closed with a Fuzzy Logic controller.

After the evaluation of the potentialities of the simulation tool, the study was extended, and GESARED was developed in order to be a generic re-entry simulator. GESARED is currently working with the model of the ARD (Atmospheric Re-entry Demonstrator). Work is in progress to introduce the model of the DART (Delft Atmospheric Re-entry Demonstrator), and another spacecraft for a Mars re-entry.

The entry phase in Earth's atmosphere corresponds to an altitude range from 120Km to close to the ground. This phase is characterized by large variations of environmental conditions. Therefore, it is necessary to have realistic models of the environment. The model of the vehicle's aerodynamic is also very important because the re-entry is characterized by variation of the atmospheric conditions and large variations of aerodynamic angles. Since the last objective is the design of control laws, the models of the sensors and actuators should also be realistic in order to obtain a good overall performance of the simulation tool.

The paper focuses on the nonlinear dynamic modeling and simulation of generic re-entry vehicles with application to the CRV and a demonstration of the ARD. In section 2, the modeling of the atmospheric re-entry dynamics is presented. In section 3, the environment is modeled. In this phase of the project, the simulated environment is Earth, but other planets will be inserted in a future phase of GESARED. Section 4 describes the modeling of the CRV. The ARD modeling is described in section 5. Section 6 shows some simulation results already obtained. The last section closes the paper with the conclusions and future work.

Planet Entry Dynamics

THE planetary entry motion is of 6 Degrees-of-Freedom (DOF). In the modeling of this motion, the 3-DOF translational motion and the 3-DOF angular motion are considered.

The translational motion represents the point-mass trajectory motion. The generic equations that describe this motion are the dynamic and kinetic equations:[2]

$$\bar{F}_E = \frac{d^2\bar{r}_{cm}}{dt^2} \quad (1)$$

$$\bar{V} = \frac{d\bar{r}_{cm}}{dt} \quad (2)$$

In order to apply the previous equations, it is necessary to choose the frames to express them, and to determine the external forces applied to the vehicle.

The external forces applied to the point mass are the aerodynamic forces, the gravity force and the thrust force. The aerodynamic and thrust forces depend on the model of the entry vehicle simulated and on the model of the entry environment. The gravity force depends on the mass of the vehicle and on the environment model. The total external force is given by:[3]

$$\bar{F}_E = \bar{F}_A + \bar{F}_T + \bar{F}_g \quad (3)$$

The choice of the coordinate system is of major importance in the modeling of the motion, since it will influence the final simulation tool. The objective is to have the minimum number of constraints, without increasing the complexity of the equations. Therefore velocity is expressed in Cartesian coordinates of north, east and down velocity. Position is expressed in spherical coordinates of latitude, longitude and distance towards the Earth Center. The final equations of translational motion are:[4]

$$\dot{v}_\delta = \frac{1}{m}F_x - 2\Omega_E v_\tau \sin(\delta) - \\ -\Omega_E^2 R\sin(\delta)\cos(\tau) - \frac{v_\tau^2\tan(\delta) + v_\delta v_R}{R} \quad (4)$$

$$\dot{v}_\tau = \frac{1}{m}F_y - 2\Omega_E(v_R\cos\delta - \\ -v_\delta\sin(\delta)) + \frac{v_\tau}{R}(v_\delta\tan(\delta) - v_R) \quad (5)$$

$$\dot{v}_r = \frac{1}{m}F_z - 2\Omega_E v_\tau\cos(\delta) + \\ +\Omega_E^2 R\cos^2(\delta) - \frac{v_\tau^2 + v_\delta^2}{R} \quad (6)$$

$$\dot{R} = -v_R \quad (7)$$

$$\dot{\tau} = \frac{v_\tau}{R\cos(\delta)} \quad (8)$$

$$\dot{\delta} = \frac{v_\delta}{R} \quad (9)$$

In these equations, the external force vector components F_x, F_y and F_z should be expressed in the Vertical

frame, with the x axis pointing North, the y axis pointing East and the z axis pointing down.

The final equations of translational motion don't present any problem, except a singularity when the latitude is $\pm\frac{\pi}{2}$. The consequence of this singularity is a constraint in the simulation tool: simulations over the poles can't be performed. However this is not a very strong constraint, since these points aren't very used in the re-entry trajectories.

The angular motion represents the rigid body attitude motion. This motion is described by the following equations:[2]

$$\bar{M}_E = \frac{d\bar{H}}{dt} \quad (10)$$

$$\bar{\omega} = \frac{d\bar{\theta}}{dt} \quad (11)$$

In order to apply the previous equations in the simulation tool, it is necessary to choose the frames where the angular velocity and attitude position are expressed. The total external moments acting in the vehicle should also be defined.

The angular velocity will be expressed in the body frame, represented by the common notation of p, q and r.

The attitude position is chosen in order to avoid singularities. If the attitude angles of roll, pitch and yaw are used, there will be a singularity whenever one particular angle is $\pm\frac{\pi}{2}$, depending on the order of rotation. This constraint is very strong when dealing with planet entry, since most of the vehicles have very abrupt attitude maneuvers. Since quaternions don't present any singularity, they will be used in the representation of the attitude position.

The total external moments acting on the vehicle are the aerodynamic moments and the thrust moments. Since the point of moment computation is the center of gravity, the moment caused by the gravity force is zero. The total moment is given by:

$$\bar{M}_E = \bar{M}_A + \bar{M}_T \quad (12)$$

The final equations for angular motion are:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = I^{-1} \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} - I^{-1} \left(\begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \left(I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \right) \quad (13)$$

$$\begin{bmatrix} \dot{Q}_1 \\ \dot{Q}_2 \\ \dot{Q}_3 \\ \dot{Q}_4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} Q_4 & -Q_3 & Q_2 \\ Q_3 & Q_4 & -Q_1 \\ -Q_2 & Q_1 & Q_4 \\ -Q_1 & -Q_2 & -Q_3 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (14)$$

In these equations, the external moment components M_x, M_y and M_z should be expressed in the Body frame, as conventionally defined for an airplane.

The dynamic model of the entry flight is completely modeled. The state vector of the simulation tool is given by:

$$R \quad \delta \quad \tau \quad v_R \quad v_\delta \quad v_\tau \quad Q_1 \quad Q_2 \quad Q_3 \quad Q_4 \quad p \quad q \quad r$$

Other state vectors could be chosen,[5] but this is the state vector that better avoids singularities without making the simulation tool very complex.

The implementation of the equations of motion in GESARED is presented in figure 1. This figure sketches a MATLAB/SIMULINK block, where the equations of motion are simulated.

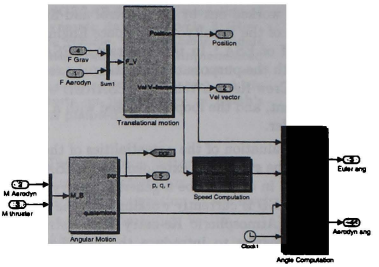


Fig. 1 Implementation of the equations of motion in GESARED

The next step in the implementation of GESARED is to model the environment. This will be performed in the next section.

Environment modeling

IN the current phase of the simulation tool, the simulated environment is Earth. The environment must be modeled in 3 characteristics: atmosphere, planet shape and planet gravitation field.

The simulation of atmosphere was performed using 2 models: Standard Atmosphere 1962 and GRAM-99.

The Standard atmosphere of 1962 was proposed by the World Meteorological Organization (WMO). The model assumes linear variation of Temperature with altitude and the division of the atmosphere in layers.[2] This is the basic relation for the construction of the atmosphere model. Other variables, as pressure and density, can be extrapolated form this relation. This model has lack of accuracy, but is very simple and allows fast computations.

The linear variation of temperature with altitude is given by the following formula:

$$T_M = T_{M_i} + L_{Z_i}(Z - Z_i) \quad (15)$$

Eleven layers were considered in this model.[3] Assuming the model of spheric Earth (only for atmospheric computations) the relations between pressure and density with altitude can be extrapolated. If $L_{Z_i} = 0$, these relations are given by the following formulas:

$$\rho = \rho_i e^{-\left[\frac{g_0(Z-Z_i)}{R_g T_{M_i}}\right]\left[1-\frac{b}{2}(Z-Z_i)\right]} \quad (16)$$

$$P = P_i e^{-\left[\frac{g_0(Z-Z_i)}{R_g T_{M_i}}\right]\left[1-\frac{b}{2}(Z-Z_i)\right]} \quad (17)$$

where $b = 3.139 \times 10^{-7}/m$.

If $L_{Z_i} \neq 0$, for the density and pressure it yields:

$$\rho = \rho_i \left[\frac{L_{Z_i}}{T_{M_i}}(Z-Z_i)+1\right]^{K_1} e^{\frac{g_0 b}{R_g L_{Z_i}}(Z-Z_i)} \quad (18)$$

$$P = P_i \left[\frac{L_{Z_i}}{T_{M_i}}(Z-Z_i)+1\right]^{K_2} e^{\frac{g_0 b}{R_g L_{Z_i}}(Z-Z_i)} \quad (19)$$

$$K_1 = -\frac{g_0}{R_g L_{Z_i}}\left[\frac{R_g L_{Z_i}}{g_0}+1+b\left(\frac{T_{M_i}}{L_{Z_i}}\right)\right] \quad (20)$$

$$K_2 = -\frac{g_0}{R_g L_{Z_i}}\left[1+b\left(\frac{T_{M_i}}{L_{Z_i}}\right)\right] \quad (21)$$

The Global Reference Atmospheric Model - 1999 Version (GRAM-99) is a model of NASA/MSFC.[6] This model is an empirical model, constituted by 3 different sub-models. The model for lower altitudes is the Global Upper Air Climatic Atlas (GUACA). This model is valid from 0 to 27 Km. For the middle atmospheric region, ranging from 20 Km until 120 Km, the Middle Atmosphere Program (MAP) data is used. Finally, the highest part of atmosphere, from 90Km until orbital altitudes, is simulated using the Jacchia model. The Marshall Engineering Thermosphere (MET) model, in this case MET-99 implements the Jacchia model. GRAM-99 is very accurate, but the computations with this model are slow.

The Earth shape model is used in the computation of altitude from the distance towards the Earth center. The model used in GESARED considers Earth as the 3-D figure obtained from the rotation of an ellipse over its smaller axis. This model is simple, but is accurate enough in the computation of altitude.

For the altitude, it yields:

$$h = \frac{-2R_E\cos(D_0) + \sqrt{(2R_E\cos(d_0))^2 - 4(R_E^2 - R^2)}}{2} \quad (22)$$

where the radius of Earth and the angle D_0 are given by:

$$D_0 = e\sin(2\delta) - \left(\frac{e}{2}\right)^2 \sin(2\delta) + \\ + 2e^2\sin(2\delta)\sin^2(\delta) + \ldots \quad (23)$$

$$R_E = \frac{R_p}{\sqrt{1-\left[1-\left(\frac{R_p}{R_e}\right)^2\right]\cos^2(\delta - D_0)}} \quad (24)$$

The Earth gravitation field model implemented in GESARED uses the Jeffery constants until J_4. Therefore the gravity force is not only dependent of the altitude, but also depends on latitude. The accuracy of this model is enough for the current applications of GESARED.

The gravity force is given by:

$$\bar{g} = \begin{bmatrix} -g_\phi \\ 0 \\ -g_R \end{bmatrix} \quad (25)$$

where:

$$g_R = -\frac{\mu}{R^2}\left[1-\frac{3}{2}\left(\frac{R_e}{R}\right)^2 J_2\left[3\cos^2(\phi)-1\right]\right] + \\ + \frac{2\mu R_e^3}{R^5} J_3\cos(\phi)\left[5\cos^2(\phi)-3\right] + \\ + \frac{5\mu R_e^4}{8R^6} J_4\left[35\cos^4(\phi) - 30\cos^2(\phi) + 3\right] \quad (26)$$

$$g_\phi = \frac{3\mu R_e^2}{R_4}\sin(\phi)\cos(\phi)J_2 + \\ + \frac{3\mu R_e^3}{2R_5}\sin(\phi)\cos(\phi)J_3\sec(\phi)\left[5\cos^2(\phi)-1\right] + \\ + \frac{15\mu R_e^4}{6R_6}\sin(\phi)\cos(\phi)J_4\left[7\cos^2(\phi)-1\right] \quad (27)$$

The Earth constants used in GESARED are given by the World Geodetic System of 1984 (WGS-84).

The environment model was implemented in MATLAB/SIMULINK. This implementation is shown in figure 2

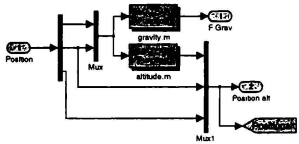


Fig. 2 Implementation of the environment model in GESARED

After the implementation of the models of the environment and the motion dynamics, the model of the vehicle should be added. In the two following sections, 2 models are discussed: CRV and ARD.

CRV

THE CRV vehicle is a NASA-ESA project and consists of a lifeboat for the International Space Station (ISS). This vehicle will be docked in the ISS, and in case of emergency, it will depart from the docking port, glide unpowered, re-enter the atmosphere, and reach a particular landing site by using steerable parafoil.[7]

The re-entry flight of the CRV is a complex process, characterized by large variations of environmental conditions, dynamic pressures, Mach number (up to 30), and angle of attack (up to 40 degrees).

The modeling of the spacecraft has 3 aspects: aerodynamic modeling, sensors modeling and actuators modeling.

The CRV is a lifting body, which produces lift by the body itself and by surface deflections. The aerodynamic database used in GESARED is from the X-38 V201, one of the CRV prototypes. The X-38 aerodynamic coefficients were measured in wind tunnel tests and are defined by a nominal value with a range of uncertain variation. A program was implemented in order to read this data and insert it in GESARED.[8]

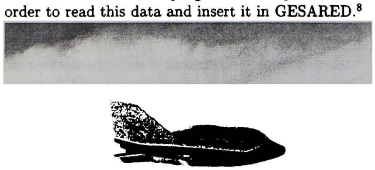


Fig. 3 X-38 Ship 2 during the 4th successful free-flight

The CRV spacecraft has 4 independently driven control surfaces: 2 flaps and 2 rudders. The rudders are classified in left and right having positive deflection when turning to the left in a rear view of the vehicle. The elevons are also classified in left and right having positive deflection when deflecting downward.

The aerodynamic computations are divided in two speed regions: subsonic / supersonic (Mach number $\leq$ 4.6) and hypersonic (Mach number $\geq$ 4). The generic aerodynamic coefficients are given by:

$$C_x = \mu_{S/H} C_{x_S} + (1 - \mu_{S/H}) C_{x_H} \qquad (28)$$

The function $\mu_{S/H}$ is the bridging function between the two speed regions. The subsonic / supersonic coefficients and the hypersonic coefficients are given by the aerodynamic table as a combination of several factors, as the deflection of the surfaces, Mach number, angle of attack, etc.

The CRV has 2 sensors that are used in the re-entry phase: the Flush Air Data System (FADS) and the Space Integrated GPS INS (SIGI).

The FADS measures velocity, angle of attack and angle of side slip. It is a modern sensor, developed for advanced airplanes flying in extreme conditions. The sensor is a parabolic probe that is mounted on the nose of the spacecraft. This probe has several holes, where pressure transducers are placed. The total pressure at these transducers is measured. The relation between the total pressure in the transducers and the measured pressure is given by a second order transfer function:

$$\frac{P_L}{P_0} = \frac{1}{\frac{1}{\omega_n^2} s^2 + 2\frac{\xi}{\omega_n} s + 1} \qquad (29)$$

A relation can be obtained between several measures of pressure and velocity, angle of attack and angle of side slip. The relation is given by:[9]

$$P_i = q_c \left[\cos^2(\theta_i) + \epsilon \sin^2(\theta_i)\right] + P_{\text{inf}} \qquad (30)$$

The parameter ϵ can be estimated by wind tunnel tests.[9] The angle of incidence is computed from the exact position of the pressure ports. The relation given in the previous equation is highly nonlinear, and therefore is solved using several estimation algorithms. In this case, the least squares algorithm was used. Other algorithms can be used, like the weighted least squares or the extended Kalman filter. The model used for the FADS was a TUDelft model of the FADS[10] and is sketched in figure 4.

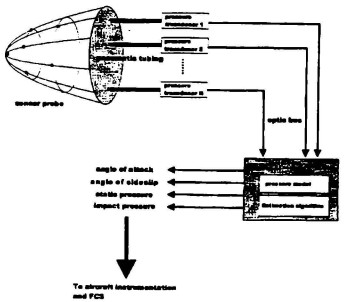


Fig. 4 Implementation of the environment model in GESARED

The SIGI provides three navigation solutions. The navigation can be Inertial, using the Inertial Navigation System(INS) system. Navigation can also

be done with Global Positioning System (GPS) satellites. The 3rd navigation solution is a combination of inertial and GPS navigation. To model the errors of the SIGI sensor, the re-entry trajectory was divided in 3 phases. The first phase starts in the beginning of re-entry until approximately 70 Km, where the combined navigation solution is used. The second phase of flight is the blackout phase, which starts at 70 Km and finishes at 30 Km. In this phase of flight the GPS is unavailable, and only the INS is used for navigation. The last phase starts at 30 Km and the combined navigation solution is used again.

In the re-entry phase, 2 actuators are used: aerodynamic surfaces and Reaction Control System (RCS).

The CRV has two types of control surfaces, i.e., the flaps (left and right) and the elevons (left and right). They are driven independently by using electro-mechanical actuators (EMA). It is difficult to build up a detailed and complete mathematical model for the actuators. Therefore, in the preliminary dynamic simulation and flight control system design, a 2nd order transfer function is used to approximate the simplified linear aerosurface deflection responses.

The RCS is a pressure regulated cold-gas nitrogen system and includes a total of 22 cold-gas thrusters. Each thruster is nominally rated at 25 lbf of thrust. Among them, 16 thrusters are located in the aft portion of the spacecraft and the other 6 thrusters are located in the nose portion of the spacecraft.

The model of the CRV was implemented in MATLAB/SIMULINK, in order to couple it to GESARED. The implementation was performed, and it is sketched in figure 5.

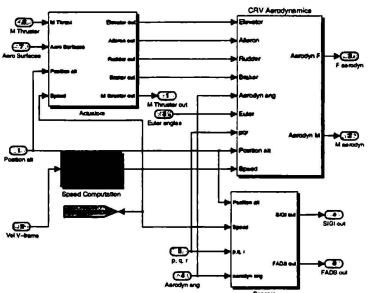


Fig. 5 Implementation of the CRV model in GESARED

ARD

THE Atmospheric Re-entry Demonstrator is an ESA demonstration project with the aim of consolidating and improving technological knowledge for the development of future reusable space transportation vehicles. The ARD (an Apollo-like shape guided unmanned capsule) is a project mainly directed to the fields of aerothermodynamics, materials, guidance, navigation and control (GNC). The flight, consisting in four phases - launch, suborbital ballistic flight, re-entry and descent - was successfully completed in the 21st October. 1998.[11]

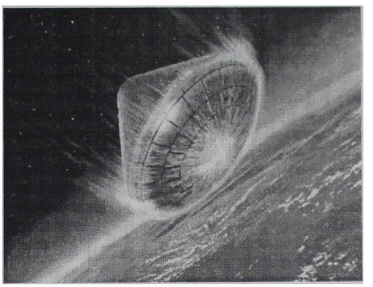


Fig. 6 Artist impression of ARD

The modeling of the vehicle for the re-entry phase can be divided in three distinct parts: sensor modelling, aerodynamic modelling and actuator modeling.

The ARD navigation system must provide velocity and positioning information that allow a landing accuracy of at least 100 km. It should be noted that there is no link between the capsule and the Ariane 5 on board computer. The navigation system comprises 2 sensors: a strapdown Inertial Measuring Unit (IMU) and a GPS receiver.[12]

The IMU outputs the position vector and the absolute velocity vector, both in the Earth inertial frame.

The GPS receiver, used to update the IMU measurements, outputs position (Altitude, Longitude, Latitude) and relative velocity (North, west, Vertical) referenced to the WGS84 geoide. The measurement accuracy, which is dependent on the altitude, is bounded by 176 and 400 m at 2 σ for the position, while for the velocity it is bounded by 2 and 2.5 m/s still at 2 σ. In case of GPS failure (blackout phase for instance) the Drag Derived Altitude (DDA) is used instead. The DDA is a mean of estimating the altitude by combining the measurements of non-gravitational accelerations, aerodynamic and atmospheric models.[13]

The aerodynamic database was obtained via wind tunnel testing and is basically divided in two parts according to the flow regime: the continuum and the free molecular regimes. To ensure a smooth transition from Free Molecular flow to Continuum flow, a bridging function, such as:

$$C_x = (1 - \mu_{RAR})C_{x_{Cont}} + \mu_{RAR}C_{x_{FMol}} \qquad (31)$$

had to be built. Since the flow regime depends on the Knudsen number, that represents the rarefaction of the atmosphere, the following error function correlation, was implemented as follows:[2]

$$\mu_{RAR}(Kn) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-y^2/2} dy \qquad (32)$$

The aerodynamic tables only provide coefficients for the longitudinal motion, however, since the ARD has an axissymmetric shape, one can consider the data tables valid for a *total* angle of attack. By *total* it is meant an angle between the axial axis (X_B) and the velocity. Thus, from any combination of α and β, 3 angles can be computed (being one of them the *total* angle of attack), with which a *total* lift force can be determined. This force will then be decomposed in pure lift (X_B Z_B plane) and side force (X_B Y_B plane).

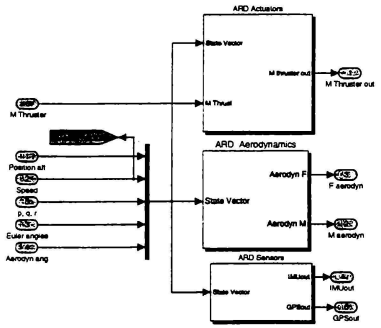


Fig. 7 Implementation of the ARD model in GESARED

In order to reach the target and to hold deceleration levels to 3.5g and thermal flux within acceptable limits throughout the whole flight, the ARD must maneuver by means of a Reaction Control System (RCS). This system, derived from the Ariane 5, is composed by 7 hydrazine 400 N thrusters positioned to generate pure moments in the 3 axes.

The implementation of the ARD in MATLAB/SIMULINK is sketched in figure 7.

Tests

THE final simulation tool was implemented, by coupling the modules of figure 1, 2 and 5. This simulation tool is sketched in figure 8.

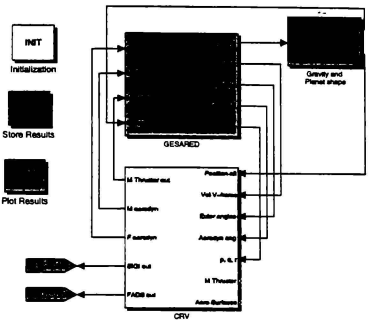


Fig. 8 Implementation of the CRV final simulation tool

Numerical tests were performed to GESARED using the CRV model. Since there wasn't a full 6-DOF trajectory available to test the simulation tool, these tests were divided in 2: 3-DOF translational motion tests and separate tests to each component of the simulation tool. In order to perform these separate tests, simplified versions of GESARED were implemented.

The 3-DOF translational motion tests were performed using 2 trajectories. The trajectories were computed by the trajectory optimization software ALTOS.[14]

The first trajectory available had the landing site at Cooper Pedy. The comparison between the output of ALTOS and GESARED are presented in Figure 9. The difference between the trajectories computed by GESARED and ALTOS is very small, and can be considered approximately the same. The small existing difference has its origin in model differences between GESARED and ALTOS.

The second trajectory from ALTOS had the landing site in White Sands. Figure 10 presents the comparison between the given ALTOS trajectory and the computed GESARED trajectory. Once again, the difference between the trajectories is very small, and can be considered approximately the same. The small existing difference has once again its cause in model differences between GESARED and ALTOS.

The entire simulator was validated separately for the full motion. Tests were performed to the CRV aerodynamic database, to the sensors and actuators, to the environment models and to the equations of motion.[3] However the full simulator for the CRV is

not yet validated, because there is no data available from a full 6-DOF re-entry trajectory that allow the results comparison.

In a first validation phase, the ARD model was tested in 3-DOF, making use of the trajectory recorded during the October 1998 flight. In this flight, the ARD took off from Kourou (French Guiana) on the top of an Ariane 5 launcher, was injected in a 5.754° tilted orbit and landed in the Pacific Ocean. The simulation itself was performed by inputting the accelerations measured during the flight, as well as the Euler angles (also in-flight measurements), in order to compute the transformation from body to vertical frame. The implemented MATLAB/SIMULINK model is shown in figure 11. The results are shown in figure 12.

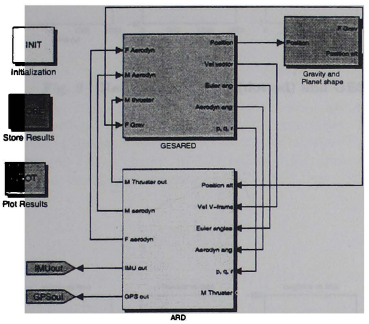


Fig. 11 Implementation of the ARD final simulation tool

From testing, it can be seen that actual flight and simulation trajectories are nearly the same, except for some minor errors, mainly due to measurement imprecisions, that are accumulated throughout the test.

Conclusions and Future work

THE re-entry simulator, GESARED, was constructed and implemented in MATLAB/SIMULINK environment. It has an open structure allowing easy extensibility and model modifications.

The 6 DOF re-entry flight dynamics and kinetics was modeled, in order to achieve the minimum restrictions and singularities. Quaternion representation of vehicle attitude was used to avoid computational singularities in the angular kinetic equations.

The Earth atmosphere was modeled based on the 1962 Standard Atmosphere Model and the Global Reference Atmospheric Model of 99. The Earth was modeled as an oblate spheroid rotating over its small axis. The gravitational potential was approximated with up to the fourth Jeffery terms in the multiple expansion of the gravitation fields.

For the CRV, fully nonlinear models were developed for the aerodynamic computations, based on data tables obtained from wind tunnel tests. The aerodynamic database was converted into the MATLAB/SIMULINK environment, and corresponding software was developed to accomplish the nonlinear computations of the CRV aerodynamics.

Simplified dynamic models of the navigation sensors, used in the CRV vehicle, were developed and implemented in the GESARED. These include FADS, Flush Air Data System, and the SIGI, Space Integrated GPS/INS.

Approximate dynamic models for the vehicle actuators were developed and implemented, which include the Reaction Control System (cold-gas nitrogen thrusters) and the aerosurface deflection actuators.

For the ARD, the aerodynamic data tables are completely implemented in MATLAB/SIMULINK environment, allowing in this manner the fully nonlinear computation of the equations of motion. The same follows for the sensors and actuators, which however, are still in implementation phase.

Having the 3-DOF simulations validated, the step ahead is to perform tests with 6-DOF, that, together with flight measurements will allow the full validation of GESARED.

GESARED is a tool in development. To make it a versatile and general simulation tool, it will be extended in some aspects. The friendly user interface is in progress. The re-entry vehicle model database is being improved; after the implementation of the CRV and the ARD, models of the DART (Delft Aerospace Re-entry Demonstrator) and a Mars re-entry spacecraft are in progress. The environment database will be improved, with the introduction of more models of Earth, and models of Mars; in the future, other celestial bodies will be introduced.

The applications of GESARED can be extended broadly. It can be a good test bed for several Guidance, Navigation, and Control (GNC) systems; currently is being used in the design of a Nonlinear Dynamic Inversion GNC system for the CRV.[15] It can also perform fully nonlinear 6 DOF simulation analysis of the re-entry vehicle.

References

[1]Ortega, G., Mulder, J. A., and Verbruggen, H. H. B., "Fuzzy Logic for Spacecraft Control: A European Approach," *Proceedings of the 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, ESTEC, Noordwijk, The Netherlands, 1999.

[2]Regan, F. J. and Anandakrishnan, S. M., "Dynamics of Atmospheric Re-Entry," AIAA Education Series, Washington DC, 1st ed., 1993.

[3]Costa, R. R., Wu, S. F., and Chu, Q. P., "GESARED - General Simulator for Atmospheric Re-Entry Dynamics," Tech. rep., ESTEC, Noordwijk, The Netherlands, May 2000.

[4]Mooij, E., "The Motion of a Vehicle in a Planetary Atmosphere," Delft University Press, Delft, The Netherlands, 1st ed., 1994.

[5]Mooij, E., "Development of START, a Six Degrees of Freedom Tool for Atmospheric Re-Entry Trajectories," ESTEC, Noordwijk, The Netherlands, 1st ed., 1991.

[6]Justus, C. G. and Johnson, D. L., "The NASA/MSFC Global Reference Atmospheric Model - 1999 version (GRAM-99)," Nasa-tm-1999-209630, NASA, Alabama, May 1999.

[7]NASA web page X-38 Technology, http://www.dfrc.nasa.gov/Projects/X38, 2000.

[8]Wu, S. F. and Chu, Q. P., "The Atmospheric Re-Entry Spacecraft CRV/X-38 - Aerodynamic Modeling and Analysis," Tech. rep., ESTEC, Noordwijk, The Netherlands, Dec. 1999.

[9]Withmore, S., Moes, T., Czerniejewski, M., and Nichols, D., "Application of a Flush Airdata Sensing System to a Wing Leading Edge," Tm-104267, NASA, Edwards, California, 1993.

[10]Tielens, P. G., "Analysis of a Flush Air Data System for High Angles of Attack Based on Wind Tunnel Measurements," Faculty of Aerospace Engineering, Delft, The Netherlands, 1st ed., 1997.

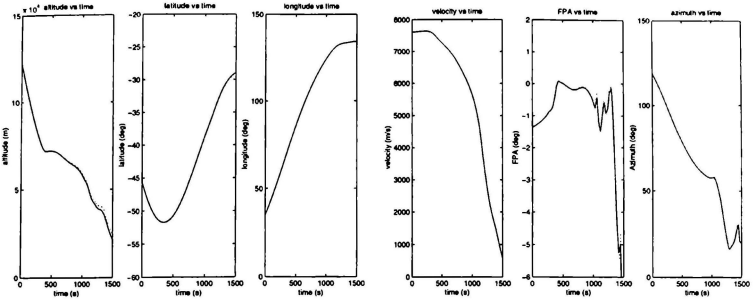
[11]ARD Post-Flight Analysis - Presentation at ESTEC, 2 1999.

[12]Bouaziz, L. and Delaux, P., "Navigation Algorithm of the Atmospheric Re-entry Demonstrator AIAA 96 3754," *AIAA Guidance Navigation and Control Conference*, AIAA, 1801 Alexander Bell Drive, Suite 500, Reston, VA 22091, 1996.

[13]Pignié, G., Clar, P., Ferreira, E., Bouaziz, L., and Caillaud, J., "Navigation Guidance and Control of the Atmospheric Re-entry Demonstrator," Aerospatiale Espace & Defence, 66 Route de Verneuil, B.P. $n°2$, 78133 Les Mureaux Cedex, France, 1996.

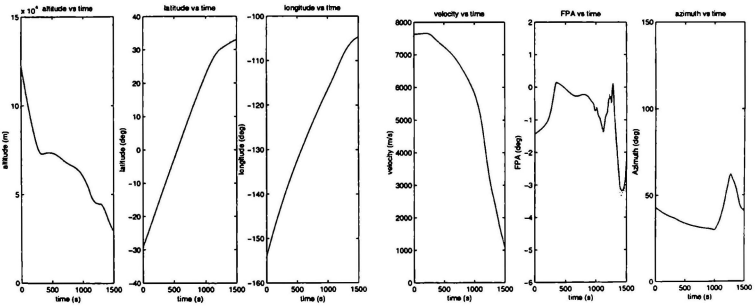
[14]Wiegand, A., Markl, A., Well, K. H., Mehlem, K., Ortega, G., and Steinkopf, M., "ALTOS - ESA's Trajectory Optimization Tool Applied to Re-Entry Vehicle Trajectory Design," *Proceedings of the 50th International Astronautical Federation Congress*, IAF, Amsterdam, The Netherlands, 1999.

[15]Costa, R. R., "GNC design using Nonlinear Dynamic Inversion for the CRV," Tech. rep., TUDelft, Delft, The Netherlands, May 2000.



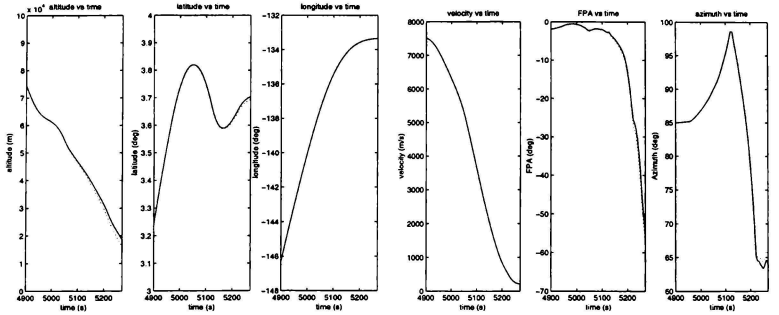
a) Position b) Velocity

Fig. 9 Results of ALTOS (dotted) and GESARED (solid) for CRV Cooper Pedy trajectory



a) Position b) Velocity

Fig. 10 Results of ALTOS (dotted) and GESARED (solid) for CRV White Sands trajectory



a) Position b) Velocity

Fig. 12 Results of real flight (dotted) and GESARED (solid) for ARD trajectory

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37224 AIAA-2000-4087

ADVANCED SPACE ROBOTICS SIMULATION FOR TRAINING AND OPERATIONS

Xavier Cyril, Gilbert Jaar, Jean St-Pierre
Space Systems, CAE
8585 Côte de Liesse, P.O. Box 1800
St. Laurent, Quebec H4L 4X4
Tel: (514) 341-2000
Email: cyril@cae.ca, jaar@cae.ca, jstp@cae.ca

This paper describes the advances being made in space robotics simulation to meet the challenges of astronaut training and space operations support. This simulator (MOTS) is being used to train International Space Station astronauts to perform on-orbit robotics tasks. It also supports mission planning and task verification in an operationally representative environment. The simulator supports critical tasks to be performed by astronauts including payload handling, berthing and de-berthing. MOTS is a state-of-the-art simulator providing astronauts with a simulation representative of the space station dynamics and visual environment. It provides real-time high-fidelity simulation of the flexible dynamics performance of two robotic arms (space station arm and shuttle arm) concurrently in a micro-gravity environment to support complex "hand-off" tasks. Contact dynamics models have been added to enhance the realism of berthing payloads to the Space Station with multiple contact points simultaneously tracked. 3D visual models support realistic views generated by the space station cameras in an operational and dynamic lighting environment that includes the production of split screen views. The incorporation of the Space Station Robot Arm Flight Control System Software provides an invaluable and confident environment in which on-orbit tasks is being planned and practiced. MOTS is also being integrated into several facilities at the Canadian Space Agency such as the Space Operations and Support Centre, to support on-line diagnostics.

Introduction

Space robotics provides a cheaper alternative to perform space systems assembly and maintenance, which otherwise would have to be performed by Astronauts during a space walk. Based on the success of the space shuttle robotic arm, the next generation robotic arm is being developed for use on the International Space Station. This arm which is being supplied by the Canadian Space Agency (CSA) will be used to perform tasks, such as, capture payloads (space station components), maneuver them out of the space shuttle payload bay and berth them to the partially complete space station. These tasks and others require dexterity in the robotic arm and considerable skill from the astronaut. The astronauts will not have direct view of the arm and its environment. They will be operating the arm from within the space station module via 3 views provided by the cameras mounted on the robotic arm and the space station. Though these cameras are an improved version of the space shuttle camera they still exhibit artifacts commonly seen in all CCD cameras.

Copyright © 2000 by CAE. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

Additionally, the 55 feet long space station robotic arm will oscillate under the combined inertia of the arm and the payload. Operating the arm under these conditions is a challenging task. In order to prepare the astronauts for this task a high fidelity simulator is needed, which will faithfully replicate the dynamics of the arm and the environment around it. Such a simulator has been developed for the CSA to support space robotic operations planning and training.

In this paper we describe the simulator, which is also referred to as MOTS. The simulator is accurately described by users as a world class simulator providing astronauts with a simulation representative of the Space Station environment through re-creation of the on-orbit dynamics and visual environment. The simulator supports the real-time high-fidelity simulation of the flexible body dynamics of two robotic arms concurrently operating in a micro-gravity environment in sufficient detail to support complex tasks. Contact dynamics models have been added to the simulator architecture to support the berthing of payloads to the Space Station with multiple contact points simultaneously tracked. Realistic views that include split screen presentation and dynamic overlays are

generated from 3D graphic models using MSS camera models to represent the operational environment. The incorporation of the Space Station Robot Arm Flight Control System Software to enhance the confidence in the simulator representation of the "as-built" flight system is currently nearing completion. Another important and challenging technical advance is the interface of the simulator with a real robot in which the simulation drives the robot through a hardware controller and contact forces are fed back to the simulation to close the loop and accurately represent contact dynamics.

Overview of MOTS

The simulator provides users with a representative environment of the Robotics WorkStation (RWS). In addition, ground support staff extensively utilize the simulator to prepare and conduct tasks under nominal and malfunction operating conditions. High fidelity software models enable the representation of flight system behavior. The simulation models provide system responses through a flight-representative human computer interface. A realistic simulation of the operating environment is also provided through the reproduction of appropriate lighting, audio, and video performance. Video images are rendered from a database of the Space Station visual objects with a customized high performance visual renderer.

The simulator supports operations analysis by providing a sophisticated set of real-time simulation and off-line data collection, conversion, and analysis tools. These tools are used to verify system operating procedures, on-board displays, telemetry definitions, system models, and performance data. The data recording and analysis function enables the user to analyze anomalous behaviour and isolate likely causes of equipment failure. The simulator also generates accurate command and telemetry data streams. These data streams are also used to support validation of the ground segment control, monitoring, and diagnostic facilities.

The main hardware components include a simulation host computer, an RWS, an instructor station, and a video distribution and recording system.

Following an extensive trade-study and benchmark investigations, the computer platform selected as the primary simulation host was a Silicon Graphics Onyx™ equipped with 12 R-4400 CPUs and 2 graphic pipes to run the visuals. This prime host system was recently upgraded to 20 R-4400 CPUs and 3 fully populated Reality Engine™ graphics pipes to enable the support of additional manipulators and camera views.

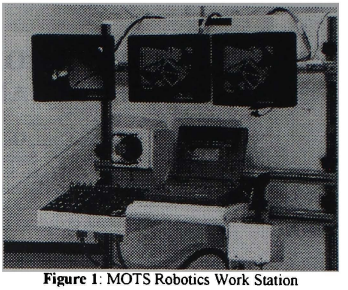


Figure 1: MOTS Robotics Work Station

The RWS, shown in Figure 1, provides crewmembers with the capability to perform all MSS commanding. This includes video system commanding and the ability to route MSS, Station, and Orbiter camera views to the three video monitors. The users interface with the simulator via on-orbit Graphical User Interface pages displayed on a laptop computer. They control the simulation via a set of 3-degree-of-freedom hand controllers (one translational and one rotational) and a display and control panel in addition to the laptop.

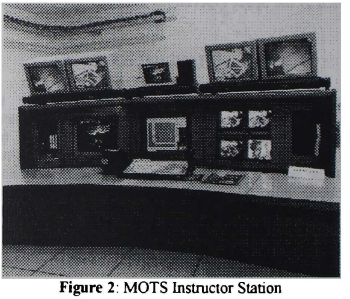


Figure 2: MOTS Instructor Station

The instructor station, shown in Figure 2 provides the capability for instructors to select, load, monitor, and control all aspects of training sessions including the ability to insert and manage malfunctions. A set of hand controllers is available to control the robotic manipulators. This station provides the instructor with an arbitrary viewpoint of the simulated scene that is positioned from a control page or by using the handcontrollers.

The simulator is built around CAE's off-the-shelf simulation environment, which provides the capability to define, build, manage and control real-time simulations and is used in flight simulators all over the world. The real time dispatcher schedules the execution of the models and processes in a specified order at a specified rate. This dispatcher handles both synchronous and asynchronous processes.

The software development environment features a suite of integrated tools to support the development of new simulation scenarios. It is integrated within the CAE simulation environment and is complimentary to the main simulation environment.

The record and playback function captures key simulator data, which enables the scene to be replayed within the simulator. It can record up to 4 channels of video and audio simultaneously on tapes, which can be replayed later either in real-time, frame-by-frame, or slow motion. The simulation can be restored to any point in the sequence, as controlled from the playback control page.

The malfunction insertion capability during a simulation session is another important function of the simulator. Specified malfunctions can be activated or deactivated manually, or based on time or event. The malfunctions are stored in a malfunction database, which contains all the malfunctions and triggering parameters.

Simulation Models

The simulation models are hosted on the computer platform and communicate with each other via the shared memory segment. In order to achieve an integration update rate of 1000 Hz, the models are parallelized across 4 CPUs. The simulation model processes are synchronized to ensure that the system is deterministic and runs in real time.

Simulation models include nominal and malfunction behavior through faithful representation of equipment performance and are divided into two major categories: physical models and equipment models.

The physical behavior models consist of a flexible multi-body dynamics model used for payload captures, berthing and handoff. Currently three robotics arms are simulated; namely the Space Station Remote Manipulator System (SSRMS), the Shuttle Remote Manipulator System and the Special Purpose Dexterous Manipulator (SPDM). The simulator also provides a power distribution model that determines the power status of the equipment, and a thermal model that computes the temperature of each MSS component based on the heat generated and dispersed.

The equipment models consist of MSS equipment hardware control models and on-board software models. One example of such a model is the simulation of the software that resides in the robotics workstation. The model provides a 2-way communication between the on-orbit control pages, control panels, and handcontrollers, and the simulated MSS elements. Commands are forwarded to the lower level equipment models and telemetry and status information is returned to the robotics workstation.

A very important addition to the simulator is the incorporation of contact dynamics that provides a realistic payload capture and berthing environment. A separate contact dynamics model supports each of the two possible types of collision scenarios. The Continuous Impulse Balance Method (or "impulse-based model"), described in [3] for a two-body system, has been generalized to support robotic systems composed of multiple flexible bodies to support general contact. The Hertz compliance method (or "Hertz-based model"), based on a toolkit provided by MD Robotics as described in [4], has been refined to accommodate the processing requirements of a real time environment and supports multiple point contact within defined berthing regions only.

Collision detection is fundamental in the overall solution of a contact dynamics problem and interacts closely with the geometric modeling of simulation objects. Two models are implemented within MOTS. The impulse-based model shares with the graphics animator the geometric database that is used to draw objects. This database is generated by a commercial graphics utility. This feature makes it possible to maintain a single repository of geometric information for all simulation bodies under configuration management. Furthermore, since what you see is what can collide, and collisions between any two (or more) objects are readily accommodated. An algorithm derived using techniques from computational geometry determines whether two objects are colliding. It returns the location of contact on the colliding bodies and the local normal to the dynamics simulation. This information is used by the impulse-based model to calculate the contact forces based on the normal approach velocity between bodies and on the estimated coefficient of restitution and duration of impact. The analysis factors in the external and control forces to allow sustained sliding contact where appropriate.

The impulse-based model is the method of choice for off-nominal, undesirable collision situations; i.e., not planned berthing sequence scenarios. The user is notified when off-nominal contact occurs and the dynamics simulation engine realistically renders the robotic system response. One drawback to this method is the limitation imposed by the accuracy with which the local normal at the point of contact can be calculated. Indeed, this is done numerically for example, by averaging the normal vectors of the polygons involved at the location of contact. When sharp edges are present the resulting normal may be such that sliding contact, which takes place in the plane perpendicular to the normal, has one body penetrating into another.

The Hertz compliance method implemented in MOTS, on the other hand, is supported by a separate geometric database of simulation objects. Contact surfaces are modeled as linear or quadratic, and collision detection correspondingly requires solving the linear or quadratic programming problem. For this approach it is necessary only to model the berthing interfaces of payloads. This method determines the interference between bodies, and uses Hertz contact law to calculate the forces. The performance of the Hertz-based model in the presence of multiple contacts and sliding interfaces is very good, and it is used whenever added fidelity in the modeling of complex berthing scenarios is desired.

Flight Software in the Loop

MOTS is required to simulate several control systems present in the MSS, at the different subsystem levels:

- Joint Control Software (JCS), the low-level joint controller
- Latching end-effector Control Software (LCS), the low-level latching end-effector controller
- SSRMS Arm Control Software (SACS), the mid-level arm controller
- Operations Control Software (OCS), the high-level MSS controller

When MOTS was initially started, the MSS flight software components were still under development. Some "engineering models" of these components, which were used for design validation, were selected for initial MOTS capability. These engineering models were at a sufficient level of fidelity to provide initial crew training.

In its long-term planning, the Canadian Space Agency had determined that MOTS should contain as much actual flight software as technically and economically feasible. The integration of actual flight software in a simulator has several advantages:

- increased level of confidence in simulator fidelity
- ease of simulator upgrade for new releases of flight software
- simulator can be used for testing flight software in complex scenarios

However, as the reader may already know, there is a cost and risk factor associated with trying to integrate any kind of real-time flight software in a simulation environment.

The first difficulty is in the reusability and modularity of the flight software code. It is generally good programming practice to develop flight software by separating it into several abstraction layers, such as a hardware interface layer, services layer, executive layer and application layer. The number of flight software layers integrated in the simulator usually reflects on the desired level of fidelity required and on technical feasibility of each option. To integrate all layers, for example, it would be necessary to provide a complete software emulation of the original processor instruction set. This is quite onerous and not necessary for an operations and training simulator (though a verification and test facility may need it).

For MOTS, it was determined that the most cost effective approach was to encapsulate the highest-level application layer into the simulator executive and host computer services. For the encapsulation of the SACS flight software, a wrapper module was provided by MD Robotics, and an interface module was developed by CAE to interface the control system with the rest of the MOTS models. This interface module is responsible for mapping data between the simulator models shared memory variables and the Ada records passed as arguments to the SACS wrapper.

The second difficulty resides in the programming language itself. The portability of the language in which the flight software is written, and the availability of a suitable compiler on the simulation platform are important factors to consider. In our case, the flight software was written in Ada. The Ada language is one of the most portable ones, since the syntax is well defined (though there are two standards, Ada83 and Ada95) and there are no assumptions made with regards to data types and data representation, which are the usual stumbling blocks for cross-platform porting.

After evaluating several compilers available for our simulation platform, it was decided to select the SGI Ada95 V1.3 compiler, which is based on GNAT 3.11b.

The main reason for choosing that solution was that the GNAT environment permitted us to easily integrate the compiler inside CAE's simulation build tool (called SIMex). This was not the case with other Ada compilers, which had proprietary mechanisms for building and maintaining executables, and which were incompatible with the SIMex configuration management functions.

Though the compilation of the SACS flight software was mostly uneventful, we did incur three run-time problems that made the simulation crash. The first problem was with the usage of "records of records". We had to modify the flight software to write into the record element-by-element instead of copying the record as a whole. The second problem was with the usage of the "use at" clause which creates a pointer to an address. Here we had to modify the flight software to introduce an intermediate variable that contained the address that was pointed to directly in the original code. The third problem was a still unexplained crash when calling one of the functions provided by the flight software itself. In this case, we had to inline the offending code in the calling program as a temporary workaround. It is expected that future compiler releases will eventually fix these problems.

Advanced Visualization

Within the Space Station the astronauts will operate the robot arm from one of two Robotics WorkStations, where camera views are displayed on three 11.4" flat panel LCD monitors. Without direct vision the astronauts will be challenged to operate the arm mainly through feedback provided by these camera views. These will tend to introduce artifacts, which make the task more demanding. The simulator generates views that are representative of the images produced by the flight cameras to support astronaut familiarization and training, or preparation for on-orbit tasks.

To generate these images in real time with the computer technology available was perhaps the greatest challenge we faced. To add realism to these images required the incorporation of shadowing and camera effects such as blooming, smearing, depth-of field, and out-of-focus. To meet these requirements CAE developed a real time visual renderer that supports all of these camera effects, the end result of that is a simulator that provides the astronauts with realistic camera views rendered at a nominal rate of 15 to 20 Hz. The visual renderer relies upon the visual database and the data provided by the software simulation models. The visual database consists of graphical models of the Space Station and MSS elements (Figure 3), constructed for efficiency and organized into several levels of detail. The database when rendered provides an impressive lighting environment that faithfully emulates that in space. The latest addition to the visual model is the capability to multiplex 2 views on the same monitor. This enables the user to have as many as 5 views on the 3 monitors of the Robotics workstation.

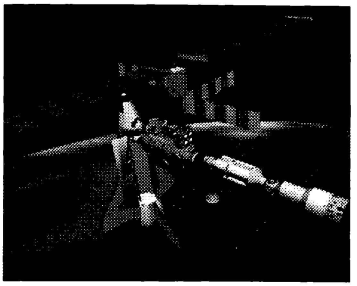


Figure 3: International Space Station

In order to run the simulation models on one platform and the visual model on another platform, a High Level Architecture protocol was implemented in the simulator. The protocol is a general-purpose architecture for simulation re-use and interoperability and is in the process of becoming an open standard through the Institute of Electrical and Electronic Engineers. This will significantly improve the visual update rate and speed up the system load time.

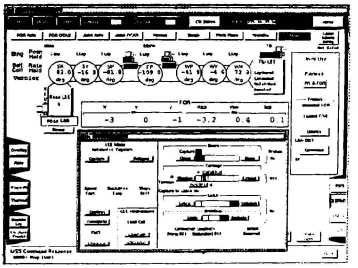


Figure 4: On-orbit HCI page

The primary method by which users interact with the MSS equipment on-board the Space Station is the

human computer interface pages (Figure 4) hosted on the Personal Computer System (PCS) laptop. The implementation and activation of these pages within the simulator was another significant challenge. To make these pages "come alive", the CAE design utilized Interface Control Documents to determine the correct scenario and derive the internal logic. Interfaces from these pages to the simulated models were determined from control system design documents. Currently the Space Station and the Orbiter arm control pages are created using an off-the-shelf package known as SAMMI, however the SPDM control pages are being implemented in C++. These pages have yet to include a real world interface based on a 1553 protocol.

Hardware in the Loop Simulation

The simulator is used to support the emerging SPDM Task Verification Facility (STVF) which embodies hardware-in-the-loop simulation. Within this facility a real robot will reproduce the Special Purpose Dexterous Manipulator end-point motion resulting from real contact forces between mock-ups of the SPDM payloads, known as Orbital Replacement Units (ORUs), and the work-site. The real contact forces will then be fed back to the simulator, which will be running the SPDM dynamics and control system models. Data will be exchanged between the simulator and the robot on a firewire, since the data exchange must be at a rate of 1000 Hz. To achieve real-time performance with the added simulation models within the available resources, a parallel simulation architecture was implemented.

The simulator will also be interfaced with the Space Operations Support Centre to drive it with simulated telemetry data. This will enable responses to simulated MSS behavioral data to be validated and compared with actual telemetry and command responses. The simulator will also continue interfacing with the Artificial Vision Unit (AVU) that locates objects by tracking appropriately positioned surface mounted targets. The AVU supports many Space Station tasks where fine positioning takes place out of the line of sight of the operator and available cameras. The simulator is also capable of supporting concurrent simulations where flight activities are shadowed in near-real time and corrected with flight data telemetry as it is received. With this capability ground support personnel will possess a powerful tool to analyze and support flight robotic tasks as they unfold.

Summary

The simulator plays a key role in preparing astronauts for the operation of the MSS and verifying operations procedures. It provides a faithful simulation of MSS equipment and its environment through high fidelity simulation models and visual simulation. The simulator is hosted within the CAE real-time simulation environment, which has an excellent track record and is proven to provide deterministic and consistent simulations in flight simulations worldwide. The simulator was used by astronauts and has gained their acceptance as a world class simulator. However, there remain features to be added in the near future which require full utilization of current technology. Additionally, the simulator will be validated against available "as built data" and ultimately against flight data. The simulator is established upon a solid technical and architectural foundation and has proven capable of supporting a variety of user needs. This capability will be enhanced with the incorporation and validation of many new capabilities in the near future.

References

1. G. Jaar, X. Cyril, D. Harvie and D. McCabe, "MSS Operations and Training Simulator", *Astro 98 Conference*, Ottawa, Ontario, 26-28 Oct 98.
2. G. Jaar, "End-to-End Simulation of the MSS within MOTS", *Symposium on Space Mission Operations*, St-Hubert, Québec, 4-5 Dec 1995.
3. F. Belanger and X. Cyril (1997) "Continuous Linear Impulse Balance Method for Real Time Simulations of Contact Dynamics", *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, AIAA Paper 97-3670.
4. M. Ou (1995), "Contact Dynamics Modeling for the Simulation of the Space Station Manipulators Handling Payloads", *IEEE International Conference on Robotics and Automation*, pp. 1252-1258.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37225 AIAA-2000-4089

VISUALIZATION, INTERROGATION, AND INTERPRETATION OF COMPUTED FLOW FIELDS – NUMERICAL EXPERIMENTS

Albert B. Hanneman, Research Analyst
Robert E. Henderson, Chief Scientist
Noesis, Inc.
Arlington, VA 22203

ABSTRACT

Complex viscous flow calculations are often presented as colored gradient plots of the flow velocity and/or pressure. There is a need to quantify the calculated data for more accurate analyses. The Virtual Windtunnel (VWT) is a system for the visualization, interrogation, and interpretation of computed flow fields that has been developed at NASA Ames. This paper discusses an application of the Virtual Windtunnel to the design of hydrodynamic vehicles. By simulating a physical laboratory experiment in an interactive virtual reality environment a given concept geometry can be evaluated earlier in the design cycle, compared to what is currently done in a physical laboratory test, since it eliminates the need to build hardware for testing in the physical laboratory. The faster turn around for the evaluation of a new concept should result in more innovative design configurations. To evaluate this approach a numerical experiment was defined and conducted to demonstrate the value of the approach. This approach is also valuable for conducting pre-test numerical experiments prior to a commitment to manufacture model test hardware and could be an integral part of a process for validating computational fluid dynamics codes.

1. INTRODUCTION

The availability of computer codes to calculate complex flow fields begs the question, are we able to accurately and effectively extract and use the information that has been calculated? Many times the results of complex viscous flow calculations are presented only as colored gradients of the flow velocity and pressure, making quantitative analyses of the calculated data very difficult. This paper describes a study to develop a means for the visualization, interrogation, and interpretation of a computed flow field that simulates the environment of an experimentalist or design expert, i.e. the physical laboratory. This allows the experience of these experts to be more readily applied to the interpretation of the calculation.

The NASA Virtual Windtunnel (VWT) software provides the ability to dynamically visualize and interrogate a computed flow field. It is dynamic in that the user can select and observe, in real time, a virtual reality environment containing any of the computed flow properties that the user desires. This study used the NASA VWT software[1] as the basis for the definition and development of a Virtual Water Tunnel and Towing Tank (VWT^3). The VWT^3 is envisioned as an application of the VWT tailored specifically to marine vehicles and naval hydrodynamic applications. A numerical experiment was defined and conducted to demonstrate the value of this concept. The test vehicle is an appended submarine hull with the calculation conducted at 2° pitch.[2] The planning and execution of the numerical experiment will be presented.

A survey and numerous site visits were conducted to help guide the project by defining the requirements of the US Navy's submarine hydrodynamic designers and experimentalists. The survey of potential users highlighted the desire of the hydrodynamics and hydroacoustics community to simulate the environment and characteristics of a physical water tunnel or towing tank test.

The uses for this software system will be in support of the process for the design of marine vehicles. It will permit earlier evaluations of a given design geometry, compared to those obtained today from physical laboratory tests. Estimations from personnel at the Applied Research Laboratory, Pennsylvania State University state that it would take between 3-4 years to develop a new autonomous undersea vehicle (AUV) concept using the traditional design process, which uses water tunnels and towing tanks to evaluate the new concept. By incorporating numerical evaluations instead of physical laboratory tests the development time would be reduced to between 12-18 months. This would result in a 63 percent reduction in development time.[†]

While calculations of the flow field associated with a new geometry will have to be conducted prior to visualization, they can be evaluated in a laboratory-like environment with which the experimentalist and designers are familiar. The VWT^3 system will allow experimentalists and designers to evaluate a computed

[†] from personal communication

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

flow field themselves, without having to rely on interpretation by the expert who performed the computational simulation. This whole process will be faster than having to build hardware for testing in the physical laboratory. The faster turn around for the evaluation of a new concept should result in more innovative design configurations.

The VWT[3] will not replace current test facilities. It will shift the focus of the test facilities from design definition, i.e. how can the geometry be modified to give the desired performance, to a role of vehicle system design verification and validation. The VWT[3] is also valuable for conducting pre-test numerical evaluations prior to a commitment to manufacture model test hardware and instrumentation. Such evaluations can aid in the identification of fluid dynamic 'hot spots' and in the placement of sensors to record critical data.

Another use of the VWT[3] is in the validation of computational codes. The dynamic comparison of experimental data and computational predictions of concept performance should provide considerable insight into the validity of a code to predict physical phenomena. This means being able to do side-by-side comparisons of predicted and experimental data.

2. NASA VIRTUAL WINDTUNNEL

The NASA Virtual Windtunnel (VWT) software uses virtual reality technology to allow the intuitive visualization and interrogation of data from modern computational fluid dynamics (CFD) simulations. The highly interactive three-dimensional nature of virtual reality allows the user to more effectively explore and analyze the complex structures found in large, unsteady, three-dimensional fluid flow simulations. The virtual environment uses a variety of input and display devices to give the user the illusion of being immersed in an interactive, computer generated environment. Different input devices (such as a mouse, joystick, wand, keyboard, VR glove, or 3D mouse) and display technologies can be used with the software depending on the application and degree of immersion required. The VWT software currently operates on the Immersive Workbench, Fakespace BOOM, CRT with head-tracked stereo goggles, and standard CRT workstation. A CAVE implementation is planned for the future.

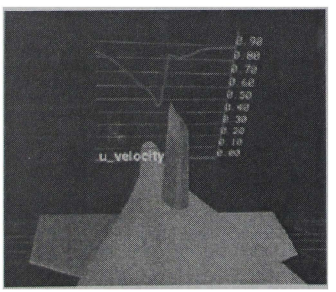


Figure 1: Two-dimensional plot of the U-component of the velocity

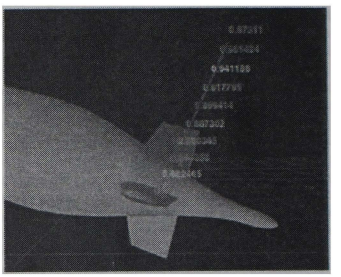


Figure 2: A rake of sample points with numerical display of the velocity boundary layer on a stern appendage.

The NASA VWT software uses an objected-orientated structure and process architecture to create an extensible, interactive visualization framework. This framework results in a high-performance visualization environment that provides the ability to use a wide variety of interface hardware and can be easily extended to include new capabilities as desired. The current version of the Virtual Windtunnel is implemented in C++ on Silicon Graphics platforms and supports a variety of interface techniques and interface hardware configurations.

During the initial design and coding of the VWT, NASA used four high level user-driven requirements to ensure that the software would be useful to the scientific visualization community.

- Extensibility: The visualization environment should be extensible in order to accommodate new visualization and interaction techniques. It should be possible to add new objects (input

devices, display technologies, and/or data analysis tools) by following a template without having to understand the entire Virtual Windtunnel code structure.[3]

- Versatility: The user should be able to configure and reconfigure the visualization environment at will. This includes adding and deleting visualization tools and control objects. The user must be able to access, and control, any portion of the data set being visualized.
- Collaborative visualization: The visualization system should support shared, collaborative visualization that allows multiple users, perhaps in disparate physical locations, to interact in the virtual environment.[4]
- User acceptance: Any new system will only be used when the benefits of using the system outweigh the difficulties and training investment. Versatility and user-friendliness are key to gaining user acceptance.[5]

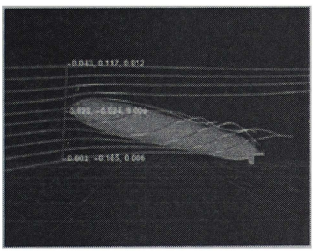


Figure 3: A rake of streamlines colored by pressure with numerical values labeling the rake coordinates in physical space.

The visualization techniques used in the NASA VWT software were inspired by the techniques used to probe the flow in a physical wind tunnel. Some of the visualization tools that are available include tufts, streaklines, particle paths, streamlines, and material tracking. These visualization tools can be controlled dynamically in real time. The user can grab, manipulate, and reposition any of the tools at any time. The VWT can display the velocity, pressure, vorticity, density, temperature, enthalpy/entropy, energy, and momentum. Additionally, these data can be displayed using streamlines, streaklines, particle traces,

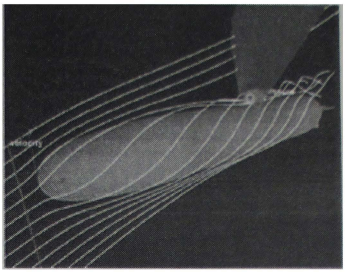


Figure 4: A rake of constant velocity streamlines and a contour plot of the pressure

numerical values, two-dimensional line plots, contour plots on arbitrary cutting planes, isosurfaces, and tufts.

The extensibility of the core VWT software was demonstrated when NASA personnel added a two-dimensional plotting capability after Noesis, Inc. suggested that was a useful visualization tool. This new data analysis tool was added to the core software in a matter of weeks, and is demonstrated in Figure 1.

Some of the visualization tools and capabilities of the NASA VWT software are shown in Figures 2 through 4. In Figure 2 a rake of sample points was created and placed in the boundary layer of a stern appendage. The numerical value of the velocity was then added to show the boundary layer. Figure 3 shows a rake of streamlines with the physical coordinates (x,y,z) of the rake attached. The streamlines are colored by the pressure. In Figure 4 streamlines of constant velocity and a contour plot of the pressure are used to show the vortex core on the top of the body.

3. THE VIRTUAL WATER TUNNEL AND TOWING TANK (VWT^3) PROJECT

The Virtual Water Tunnel and Towing Tank was proposed as a means to study and interpret computed flow fields in an environment that simulates the hydrodynamic laboratory facilities currently used. This would allow the user to view the results of CFD calculations in an environment with which they are most accustomed, yielding more effective design decisions and better designed experiments. When operated over the distributed area network installed at the Navy's Hydrodynamic/Hydroacoustic Technology Center (H/HTC) the VWT^3 would enhance the effectiveness of, and collaboration between,

hydrodynamic designers, experimentalists, and computational fluid dynamics (CFD) experts.

The specific goals and objectives of the project are as follows:

1. ***Simulate existing hydrodynamic test facilities*** – The VWT^3 will be used to simulate existing experimental laboratory facilities. The data presented in the simulated environment should visually resemble that measured in an experimental laboratory facility to provide the user with an environment conducive to the decision process currently used during a laboratory experiment.
2. ***Provide enhanced evaluation and interpretation of numerical data*** – Current methods for evaluation and interpretation of CFD data rely primarily on static two-dimensional analyses. The VWT^3 will be used to interactively and dynamically visualize both two and three-dimensional CFD data sets. The use of a virtual environment will permit the user to view the data with powerful tools in a more intuitive and "hands-on" fashion than presently possible, resulting in increased productivity.
3. ***Conduct pre-test numerical experiments*** – The ability to provide enhanced evaluation and interpretation of data will allow the VWT^3 to be used to conduct pre-test numerical experiments by visualizing and interrogating a computed simulation of an experimental test, including any facility interference effects. This allows the experimentalist to examine the expected flow field and fine tune the experiment before committing to a particular test plan and hardware build. A pre-test experiment will allow experimentalists to identify specific regions of interest more precisely than may be possible using just experience. Simulated sensors could also be used to aid in the placement and sizing of sensors and probes used in the physical experiment, resulting in more meaningful experiments being conducted.
4. ***Facilitate tool validation*** – The VWT^3 will allow both computational and experimental data to be viewed, and compared, simultaneously. This will allow design and analysis tools to be both qualitatively and quantitatively validated.
5. ***Provide a training tool*** – The VWT^3 will enable more comprehensive training of both experimentalists and designers. The numerical simulation of a laboratory experiment can be used to train experimentalists at far less cost than performing the same experiment in a physical facility. An instructor could also discuss sensor placement and demonstrate the benefits of one sensor over another, which would not be possible in a physical facility without conducting several experiments.
6. ***Enhance collaboration*** – A collaborative virtual environment provides for real-time, interactive data exchange. Integrating the VWT^3 into a collaborative environment will facilitate data communication and interaction between project team members at different physical locations. A collaborative environment would allow a diverse group of experts in different fields, such as propulsion and/or acoustics and/or maneuvering, to interact and influence the design from their technical perspectives without requiring their presence in the laboratory.

4. VWT^3 SURVEY

To ensure that the VWT^3 will be an effective tool, we asked for input from potential users of the system in order to define and document their requirements for conducting numerical experiments and evaluating data in an immersive environment. We felt that user input during the system definition phase of the project is important because it will help ensure that the Virtual Water Tunnel and Towing Tank will be a useful tool for as many different users as possible. The first step in getting input from potential users was a User Survey, which was distributed to members of the Navy's hydrodynamics and hydroacoustics community.

The objective of the User Survey was to define and document the requirements that potential users had for conducting numerical experiments and evaluating data in an immersive environment. The survey was also used to determine what computational data analysis tools were currently being used, and what capabilities the potential users would like to have incorporated into the VWT^3.

As a follow up the User Survey, we visited selected organizations to discuss and get feedback on the project. The organizations visited were the Naval Undersea Warfare Center (Newport, RI), the Naval Surface Warfare Center (Carderock, MD), and the Applied Research Laboratory at Pennsylvania State University. We met with approximately 39 designers, experimentalists, and CFD experts from the different organizations, and these discussions, along with the User Survey, helped us define the system requirements for this project.

5. VWT^3 SYSTEM REQUIREMENTS DOCUMENT

After completing the VWT^3 survey and associated site visits a list of top-level system requirements was created. This list, along with a survey of the state-of-the-art in visualization hardware and software, was used to write a System Requirements Document, which

described the characteristics and capabilities of the required system to visualize, interrogate, and interpret incompressible, hydrodynamic pressure and velocity fields for naval applications. The NASA Virtual Windtunnel will be the core software around which the Virtual Water Tunnel and Towing Tank system will be developed. The VWT^3 will build on the VWT software in order to adapt it to the unique requirements of the naval hydrodynamic and hydroacoustic community.

The system requirements for the VWT^3 were based on the needs and input from four different categories of potential users. These users are designers, performance analysts, experimentalists, as well as computational experts.

5.1 DESIGNER

Integrating numerical experiments into the design practice will allow an early evaluation of advanced concepts without having to building a physical model and other test hardware. Then, given a design scenario the user can:

- visualize and interpret different design iterations data side-by-side, or by superposition, to facilitate design decisions (flow lines, velocity and pressure distributions, forces, etc.)
- compare current geometry with previous designs
- interpret a computed flow field and then define the next design cycle configuration

5.2 PERFORMANCE ANALYST

Given a computed time-mean and unsteady velocity and pressure fields in real time the user can:

- identify deficiencies in computational grid construction
- calculate forces from surface pressure and shear stress distributions
- comparison surface flow & wall shear stress distributions
- display fluid dynamic & potential energy distributions
- display linear & angular momentum distributions
- compute fluid vorticity in a user specified region
- conduct Fourier & Hilbert harmonic analyses of spatial data
- inject particles into the flow field to determine origin of non-uniformities
- track vortices to determine areas with significant fluid/ structure interactions

5.3 EXPERIMENTALIST

Prior to conducting a test the user can:

- identify "hot-spots" or regions of high gradients
- determine anticipated levels of quantities to be measured
- evaluate sensor placement to capture desired quantities

Given an experimental data set the user can:

- interactively and dynamically view the measured data set in a three dimensional environment
- conduct analyses to interpret the data, e.g., Fourier & Hilbert harmonic analyses of spatial data
- visualize experimental and computed data side-by-side and by superposition to facilitate code validation

One of the conclusions resulting from the VWT^3 User Survey was that any visualization system must be able to operate in both full immersive and non-immersive modes. While most of the potential users would use the immersive environment to perform numerical experiments, they would prefer to conduct the majority of their work at their personal computer/workstation. Therefore the VWT3 must be flexible and have full functionality in immersive, semi-immersive (workstation with stereo glasses), and non-immersive modes.

6. NUMERICAL EXPERIMENT

A numerical test plan was created to document the operational aspects of different visualization systems. The test plan simulated a laboratory test and defined a standardized process for evaluating and verifying the performance of scientific visualization software. A typical Navy data set, in this case a Mississippi State calculation of the flow around the DARPA SUBOFF model using the UNCLE code, was used as the basis of the experiment. [6,7]

The numerical experiment was performed on several different visualization software systems in order to evaluate the operational aspects of each system. While the different systems were developed to satisfy different sets of requirements, we were evaluating the relative performance of each system in order to determine which system came the closest to meeting the VWT^3 requirements. Out of the three systems we evaluated using the numerical experiment the NASA Virtual Windtunnel software system was by far the most mature, and in fact was the only one to successfully complete the numerical experiment.

There were three main objectives of the numerical experiment:

- ✧ To evaluate the applicability of the NASA VWT to simulate existing Navy hydrodynamic and hydroacoustic test facilities
- ✧ To provide a basis (i.e. data) to evaluate the transition of the VWT to the Navy's H/HTC
- ✧ To gain on-site experience in the use of the VWT software and associated hardware systems

Once each of these objectives are met it will be possible to evaluate the capabilities of the software system.

Before conducting the numerical experiment the person conducting the experiment became familiar with the VWT software by reading all the available documentation, both printed user's manuals and documentation available on-line at the NASA web site. The documentation provides a general overview of the system, explains each of the visualization tools, and includes a Quick Start tutorial.

6.1 METHODOLOGY

The numerical test plan follows the same flow of steps that is used in an experimental test. In the following these steps are shown in Italics, along with a description of the corresponding numerical activity. The purpose of a laboratory experiment is generally to evaluate specific design geometry at specific operating conditions. The purpose of the numerical experiment described in this test plan is to output specific data for evaluation of a particular flow field calculation corresponding to a specific operating condition.

(1) *Build the model:* The model geometry that will be used in the numerical experiment is the same geometry that was used to build the wind tunnel model for the DARPA SUBOFF experimental program. The geometry is described by a set of equations, which give either the axial and radial values for the axisymmetric components or the Cartesian coordinates for non-axisymmetric components.[8] Computational fluid dynamic (CFD) calculations (supplied by Dr. David L. Whitfield of Mississippi State University using their UNCLE RANS code) of the flow field around the SUBOFF model will serve as the benchmark for the numerical experiments. Three different configurations will be tested to study a range of different conditions, similar to those seen in an experimental facility.

   a. bare hull @ 0 degrees pitch
   b. bare hull @ +2 degree pitch
   c. fully appended hull @ 0 degree pitch

(2) *Model installation and shakedown:* The SUBOFF model will be "installed" in the virtual environment by loading the CFD data files into the computer system. This means that the SUBOFF geometry file and the calculated flow field will be used as input for the NASA Virtual Windtunnel software. Once all the data files have been loaded into the system, a visual assessment of the calculated flow field will be conducted by looking at some streamlines around the model. This is to be a quick check of the flow field to ensure that the data files were loaded into the system properly, the numerical model was "installed" correctly, and that everything is working properly.

(3) *Data acquisition:* A detailed evaluation of pre-specified parameters, as visualized in the VWT, will be made. Specific quantities and data will be displayed and recorded (measured) using the NASA software so that they can be compared with the original MSU data set to verify data accuracy within the virtual environment. These "measurements" will be made by recording specified parameters at prescribed locations in the flow field. To simulate the dynamic and interactive nature of an experimental test the "measurement" locations are determined during the course of the numerical experiment. The user will not know the specific measurement locations a priori, but will determine them during the experiment based on previous measurements.

   There are only three quantities that can be directly measured in a wind tunnel experiment. These flow quantities are velocity, pressure, and temperature. From these three quantities many of the other important parameters that are used in the evaluation process can be derived. Some of these other parameters are vorticity, helicity, shear stress, mass flow, density, lift, and drag. However, since the purpose of this test plan is to simulate an incompressible hydrodynamic laboratory experiment, only the velocity and pressure will be "measured" during the numerical experiment and the temperature is assumed to be constant.

(4) *Data reduction:* After the "measurements" have been made it will be necessary to compare the velocity and pressure measurements with the original Mississippi State SUBOFF data set and access the accuracy of the software. This will be done using standard x-y plots including both "measured" and computed values on the same plot. In addition, contour plots of the difference between "measured" and computed values will be created to show a more global view of the flow field. These contour plots will be created for each of the velocity and pressure "measurements" taken in the data acquisition phase of the test.

(5) *Data Interpretation:* Different quantities will then be compared between the values "measured" during the virtual experiment and those straight out of the computed data set.

   a. surfaces of constant velocity, pressure, and density (which should be constant across the entire flow field)
   b. streamlines
   c. surface streamlines
   d. streamwise vorticity

6.2 RESULTS

The NASA Virtual Windtunnel software was successfully used to perform a numerical experiment as outlined in the numerical test plan. The VWT software was used to conduct the numerical experiment in an immersive 3D environment, in 3D stereo workstation mode, and on a 2D CRT workstation. The three-dimensional immersive environment was created using an Immersive Workbench from Fakespace Systems Inc. with stereoscopic head tracked glasses and a motion tracked wand device. The three-dimensional stereo workstation used stereoscopic head tracked glasses and a standard mouse. All three visualization environments used the exact same version of the Virtual Windtunnel software with the only difference being different configuration commands in the start-up script.

The Virtual Windtunnel software reads in any standard *plot3d* data file. Before conducting the numerical experiment it took a few minutes for the NASA personnel to modify a script file to read in the Mississippi State data *plot3d* file. Once the script file was created the data could be viewed on the system.

Approximately 1.25 hours were spent being "trained" by the software developers in using the VWT system in order to become familiar with the visualization tools and the immersive workbench. In general, the system is very intuitive and easy to use and not much training was required. The VWT uses pull down menus and slider control bars to choose, modify, and manipulate the visualization tools.

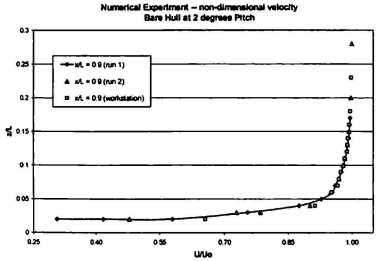


Figure 5: Velocity profile data from the numerical experiment

As part of the "training" streamlines were generated and a quick check of the flow field was performed to make sure the data file was being read correctly. This shakedown period included making a visual evaluation of the streamlines, pressure field, and flow structure verified that the numerical model was correctly "installed" in the virtual wind tunnel.

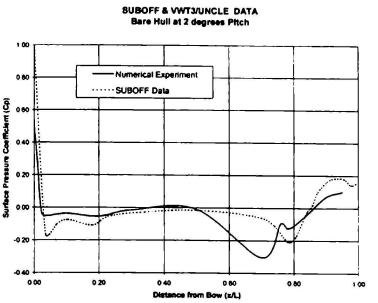


Figure 6: Surface pressure data from the numerical experiment plotted against the SUBOFF experimental data

After the "training" period and shakedown we proceeded with the numerical experiment. Boundary layer velocity profile measurements were made along the hull ($x/L < 1.0$) and wake surveys were taken behind the body ($x/L > 1.0$). This was done for several velocity components at different axial, radial, and circumferential locations. The measurements were made by creating a rake of sample points and recording the velocity and physical location in space at each point. Flow field pressure measurements were then made at different spatial locations around the body and static pressure measurements were made on the surface of the body.

Figure 5 shows three velocity profiles measured during the numerical experiment. The first two velocity profiles were measured when using the Immersive Workbench, while the third profile was measured when conducting the virtual experiment on a workstation equipped with stereoscopic headtracked glasses. In general the agreement is quite good and there was no difference when taking measurements in the immersive environment as compared to using a standard workstation environment.

Figure 6 shows surface pressure data measured on the top dead center of the model during the numerical experiment compared to data measured during the SUBOFF experiments. The pressure coefficient is defined as

$$C_P = \frac{p - p_0}{\frac{1}{2}\rho V^2} \quad (1)$$

where p is the measured pressure, p_0 is the freestream pressure, V is the velocity, and ρ is the density. Figure 6 shows that the numerical experiment does not accurately capture the pressure field around the body. However, it must be mentioned that the numerical experiment is only as good as the data from which the visualization is derived. If the computational fluid dynamics code used to generate the numerical "model" does not accurately predict the flow field the numerical experiment can not accurate duplicate the physical experiment. Thus, a visualization system like the VWT can be used to quickly show areas where CFD simulations have difficulties predicting physical phenomena.

6.3 LESSONS LEARNED

Several limitations of the NASA Virtual Windtunnel were identified during the numerical experiment. The first limitation was that the VWT only displayed numerical values to two significant digits. This affected the accuracy of the measurement and the repeatability of the experiment since all numerical values, both the parameter values (velocity and pressure) and the physical location (x,y,z) in space, were displayed using only two significant digits. However, after the numerical experiment was finished NASA personnel modified the software and changed the display of numerical values to include three significant digits. Making this modification took approximately 5-10 minutes. By increasing the accuracy in presenting the data it is anticipated that the measurements, as shown in Figure 2, would be much more accurate.

Another limitation was the lack of a method for outputting numerical data. During the numerical experiment the data (velocity, pressure, and physical coordinates) had to be recorded by hand. This was a time consuming procedure, a method to automatically output numerical data was something that the Virtual Windtunnel designers had not thought to include. Since the numerical experiment Neesis, Inc. programmers have modified the core VWT code to include a subroutine for outputting numerical values into a data file.

7. CONCLUSIONS

In a speech to the International Test and Evaluation Association Symposium Paul Kaminsky, then undersecretary of defense for acquisition and technology, talked about the New Zealand victory in the 1992 America's Cup, which was a 5-0 whitewash of the American team.[‡] He talked about how Team New Zealand reinvented the traditional yacht design, which relies upon testing scale models in water towing tanks and wind tunnels. Instead, Team New Zealand integrated the yacht designers, testers, and sailing crew into a team and used a simulation-based process of design, analyze, test, evaluate, and redesign. In doing this they gained a competitive advantage by developing a superior capability, affordably and in less time than their competitors. Similar advantages could be found by using numerical experiments conducted inside an immersive virtual environment to evaluate new design concepts instead of conducting physical experiments in water tunnels and towing tanks.

The main purpose of the described effort was to verify that a numerical experiment could be conducted inside an immersive virtual environment. The numerical experiment should have the same "feel" as conducting the physical experiment in a water tunnel or towing tank. The results of this study indicate that the NASA Virtual Windtunnel is a dynamic, interactive visualization environment that can be used to conduct a numerical experiment. In addition, the software can be used to allow remote users at different geographic locations to collaborate inside the virtual environment.

However, the Virtual Windtunnel is only a visualization system, therefore any numerical experiment conducted using it will only be as good as the data set being used as the "model". This means that as the computational codes get more and more accurate the accuracy of the numerical experiment will also increase.

Another benefit of using the Virtual Windtunnel software to conduct a numerical experiment is that it allows the entire data set to be visualized, rather than just a small segment of the data. Current computational fluid dynamic simulations generate a tremendous amount of data, which can be very hard to deal with. Most of the CFD generated data is currently being analyzed using two-dimensional visualization packages. However, with the large size of modern CFD data sets only part of the data is really studied. Slices of the data where interesting or important phenomena are thought to take place are studied. By using an immersive visualization system such as the Virtual Windtunnel the entire data set can be studied, which allows the user to get a better feeling for the complete flow field.

[‡] International Test and Evaluation Association Symposium, Huntsville, Alabama, October 3, 1995.

ACKNOWLEDGEMENT

This work was conducted under Naval Sea Systems Command, SEA 93R contact N00024-99-C-2901, Stan Krycinski, Technical Program Manager. The authors would like to thank Steve Bryson, Bryan Green, and Dave Whitney of NASA Ames for their help with the Virtual Windtunnel software and Dr. David Whitfield of Mississippi State University for providing the numerical data.

REFERENCES

1. Bryson, S. and Gerald-Yamasaki, M., *The Distributed Virtual Windtunnel*, **Proceedings of Supercomputer '92**, Minneapolis, MN, 1992.

2. Huang, T., Liu, H.L., Groves, N.C., Forlini, T.J., Blanton, J.N. and Growing, S., *Measurements of Flows over an Axisymmetric Body with Various Appendages (DARPA SUBOFF Experiments)*, **Proceedings, 19th Symposium on Naval Hydrodynamics**, pp. 321-346, National Academy Press, Washington, DC, 1994.

3. Bryson, S., Johan, S., and Schlect, L., *An Extensible Interactive Visualization Framework for the Virtual Windtunnel*, **Proceedings of the Virtual Reality Annual International Symposium 1997**, Albuquerque, NM, 1997.

4. Amundson, Holly, *Distance no barrier in distributed, collaborative Virtual Wind Tunnel*, **NASA Gridpoints**, Winter 1999, pp. 23-26. *http://www.nas.nasa.gov/About/Gridpoints/gridpoints.html*

5. Bryson, S., Johan, S., Globus, A., Meyer, T., and McEwen, C., *Initial User Reaction to the Virtual Windtunnel*, **AIAA 95-0114**, 33rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 1995.

6. Huang, T., Liu, Han-Lieh, Groves, Nancy C., *Experiments of the DARPA SUBOFF Program*, **DTRC/SHD-1298-02**, December 1989.

7. Haussling, H.J., *Transition and Testing at CDNSWC of the UNCLE.6DOF Maneuvering code*, **CRDKNSWC/HD-1318-02**, November 1997.

8. Groves, N.C., Huang, T.T., Chang, M., *Geometric Characteristics of DARPA SUBOFF Models (DTRC Model Nos. 5470 and 5471)*, **DTRC/SHD-1298-01**, March, 1989.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37226

AIAA-2000-4090

SIMULATION OF AIRCRAFT LANDING GEARS WITH A NONLINEAR DYNAMIC FINITE ELEMENT CODE

Karen H. Lyle*, Karen E. Jackson†, and Edwin L. Fasanella‡
US Army Vehicle Technology Directorate, Hampton VA 23681

ABSTRACT

Recent advances in computational speed have made aircraft and spacecraft crash simulations using an explicit, nonlinear, transient-dynamic, finite element analysis code more feasible. This paper describes the development of a simple landing gear model, which accurately simulates the energy absorbed by the gear without adding substantial complexity to the model. For a crash model, the landing gear response is approximated with a spring where the force applied to the fuselage is computed in a user-written subroutine. Helicopter crash simulations using this approach are compared with previously acquired experimental data from a full-scale crash test of a composite helicopter.

INTRODUCTION

An important aspect of crashworthiness research is the demonstration and validation of analytical/computational tools for accurate simulation of airframe structural response to impacts. Crash simulation codes can be used during the airframe design phase to certify seats and aircraft, to predict seat and occupant response to impact with the probability of injury, and to evaluate numerous crash scenarios not economically feasible with full-scale crash testing.

Previously, rotorcraft and aircraft crash simulations were performed using semi-empirical kinematic codes such as KRASH [1]. The aircraft model typically contained less than 100 elements and was comprised of lumped masses, beams, and springs with user-defined crush properties based on experimental data. These models required significant engineering effort to reduce a complex aircraft structure to a simple mass-beam-spring model. Such models can provide significant information regarding overall aircraft dynamics. However, these models are not sufficient for detailed structural analysis. In addition, significant component testing is required to generate experimental data to describe the structural crush characteristics of the springs.

Technology advances have enabled the simulation of aircraft and spacecraft crashes in a reasonable time using detailed finite element models. These models can be executed in a nonlinear transient dynamic finite element code, such as LS-DYNA [2], MSC.Dytran [3], and PAM-CRASH [4]. These codes are commercial products based on the well validated, public domain code DYNA3D developed at Lawrence Livermore National Laboratories. Although the duration of an actual crash may be several seconds, the significant impact duration is typically less than 200 milliseconds. The CPU time of the simulation is dependent on the time step, the model size, and the total simulation time. CPU times from one day to one week on a high-end workstation are common to simulate the 200-millisecond impact response of an aircraft.

Many aircraft now feature energy absorbing landing gears as part of the crash energy mitigation system. The accuracy of the prediction of the fuselage crash accelerations is directly dependent on the accuracy of the fuselage impact velocity and attitude. Therefore, any velocity reductions or changes in impact attitude resulting from the landing gear stroking must be modeled. However, the accelerations during landing gear stroking, but prior to fuselage impact, usually do not cause human injury or significant structural damage. For this reason, a method to model the landing gear effect without adding substantial

---

* Aerospace Engineer
† Aerospace Engineer, Associate Fellow
‡ Aerospace Engineer
This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

complexity to the finite element model is needed. This paper highlights such a methodology.

The development of an innovative landing gear model for implementation in a nonlinear transient dynamic finite element code is described. The landing gear model was combined with a rigid rotorcraft fuselage model for the simulations presented in this paper. The results from the simulations are correlated with previously acquired experimental data.

DESCRIPTION OF TEST

The Sikorsky helicopter flight test article, developed under the U. S. Army's Advanced Composite Airframe Program (ACAP), see Figure 1, was recently crash tested at the NASA Langley Research Center Impact Dynamics Research Facility (IDRF), see Figure 2. This full-scale helicopter was designed and constructed in the 1980's to evaluate the use of composite technology. The resulting ACAP airframe consisted of 82% composite materials, and the total weight and cost savings achieved based on the final design were 23% and 24%, respectively. A systems approach was used in designing the helicopter for maximum crash protection, including energy absorbing landing gear, crushable subfloor structure, and load-limiting seats. However, the primary energy absorbing elements were the landing gear. The landing gear were designed to remove 80% of the energy for a vertical drop performed at the impact conditions of 38-ft/s, 10° pitch, and 10° roll as specified in Reference [5]. The main landing gear contained an aluminum honeycomb tube that would dissipate kinetic energy during a crash event through stable crushing, with a maximum crush stroke of 18 in.

The full-scale crash test of the helicopter was conducted on June 22, 1999 at the IDRF. Details regarding the helicopter, test conditions, instrumentation, etc., can be found in Reference [6]. The helicopter was suspended from the IDRF and dropped such that the impact attitude was 6.25 degrees pitch (nose-up) and 3.5 degrees roll (left), and the impact velocity was -462 in/sec (-26 mph) vertical and -384 in/sec (-21.8 mph) horizontal. The pendulum nature of the drop test induced a rotational pitch velocity of 9.6 °/sec (nose-up).

Onboard hardware included four energy-absorbing seats with anthropomorphic dummies. Approximately 90 channels of data, including 62 accelerometers, were collected at a 10 kHz sampling rate using a digital data acquisition system. A schematic drawing of the helicopter indicating the locations of selected accelerometers used in the analytical/ experimental correlation is shown in Figure 3. In general, high quality experimental data were obtained from the crash test. All acceleration data from the crash test were analyzed and checked for polarity errors, zero-offsets, and noise. Acceleration data in the vertical direction were integrated to obtain the vertical velocity change. The integration also provided a quality check of the data. Those channels in which the integrated velocity change varied greatly from the nominal vertical impact velocity were not used for the correlation with the analysis.



Figure 1. Photograph of ACAP helicopter at time of impact.

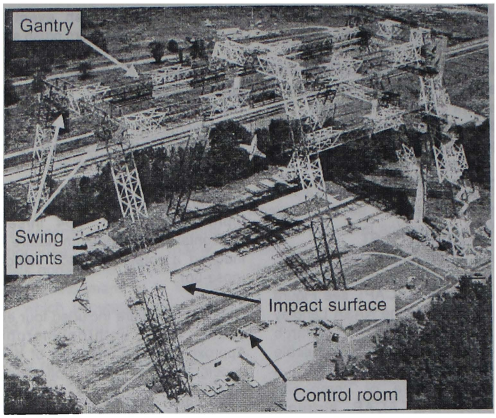


Figure 2. Schematic of NASA Langley Research Center Impact Dynamics Research Test Facility.

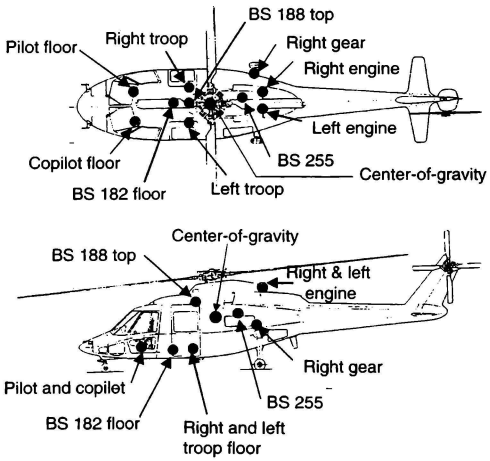


Figure 3. Schematic of selected accelerometer locations.

DEVELOPMENT OF CRASH FINITE ELEMENT MODEL

A two-stage modeling approach was used to conduct the crash simulations using the nonlinear, transient dynamic, finite element code, MSC.Dytran. For impacts on hard surfaces, accurate simulation of the energy absorption behavior of the landing gear is imperative to accurately predict the impact response of the fuselage. The landing gear stroking not only reduces the fuselage impact speed, but can also change the impact attitude. The stroking of the landing gear, which can typically last 100 milliseconds, generally provides low acceleration levels, and thus insignificant elastic deformations to the fuselage, as compared to the fuselage impact event. These facts enabled utilization of a rigid fuselage model during the landing gear stroking. Prior to fuselage contact, the nodal velocities and positions from the rigid simulation are then input into a flexible model. Correlations of the flexible fuselage model simulations with the experimental data are presented in Reference [7]. The time step for rigid models is typically an order of magnitude larger than that for the flexible model, therefore the required clock time to perform a simulation is reduced by an order of magnitude. For the model presented here, the CPU time was reduced by a factor of eight. In addition, the rigid model made the introduction of the pitch angular velocity much easier.

Landing Gear

A schematic of the ACAP main landing gear as viewed aft of the aircraft is shown in Figure 4. The main gear were designed with a two-stage, energy absorption approach. For landings within the normal operational range, an oleo-pneumatic energy absorber has been incorporated. For severe or crash landings, additional energy may be absorbed with the stable crushing of an aluminum honeycomb column within the gear. The transfer from the oleo-pneumatic to the honeycomb stage is accomplished by shear pin failures based on a predetermined force. The orientation of the gear with respect to the fuselage remains nearly constant while the oleo-pneumatic stage is stroking. The angle with respect to the vertical, as shown in the figure, is 11.8 degrees. As the honeycomb crushes, the gear rotates outward an additional 20 degrees. The drag beam controls the gear rotation.

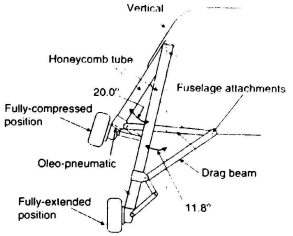


Figure 4. Schematic of ACAP landing gear looking aft.

To simplify the main landing gear mechanism as much as possible in the finite element model, the angle of the landing gear was fixed with respect to the aircraft vertical for the simulations. Therefore, the translational and rotational motions of the gear have been replaced with purely translation motion. The angle was determined by bisecting the angle through which the gear strokes, or 11.8° + 0.5 x 20° = 21.8°, as shown in Figure 5. For a crash model, the landing gear force response can be approximated with a spring where the force is computed in a user-written subroutine. The "spring" force is dependent on the relative displacement and on the relative velocity of the connected nodes. Initially, it is assumed that the gear is fully deployed, as would be expected in a crash scenario.

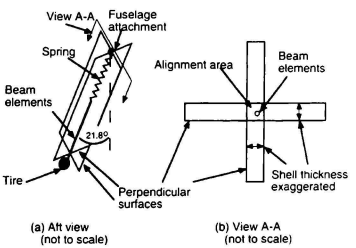


Figure 5. Schematic of landing gear in FEM.

The original crashworthy nose gear had been removed and replaced with a non-crashworthy standard nose gear. Modifications were required to make the existing nose gear more crashworthy. The modified nose gear was modeled as a spring having a constant spring force of 8,000 lb. to represent the crush strength of the honeycomb-filled aluminum tube that was inserted inside the gear.

Alignment of the gear spring relative to the aircraft can be a challenging modeling problem. A number of rigid joints, such as sliding, rotational, ball, and universal joints, are currently available as standard capabilities in commercial codes. However, these joints, which have only a small number of internal nodes (e. g., three), become unstable when large forces are applied - such as those experienced by the landing gear during a severe impact.

For this reason, these built-in joints were replaced in the model with a component containing several nodes and beam elements over which the forces are distributed, see Figure 5(a). Nodal alignment was accomplished by creating four contact surfaces on the two perpendicular plates. Each perpendicular plate was modeled with a rigid quadrilateral shell element. Rigid beams were used to connect the top rigid shell nodes to the fuselage at the attachment point. The spring force was defined in a user-written Fortran subroutine. The lower section of the gear was modeled with flexible beams attached to concentrated masses representing the tire. Each main gear model consisted of 40 nodes and 39 beam elements. The nose gear model consisted of seven nodes and six beam elements. The gear nodes were then constrained to remain within the intersecting region of the perpendicular shells, see Figure 5(b). The thickness of the alignment shells was set to provide sufficient stability without creating extremely large contact forces. For the results presented in this paper, the shell thickness was 0.010 in.

For simplicity of model development, the initial landing gear models were attached to a simple triangular rigid plate element, which approximated the fuselage. The aircraft mass, center of gravity, and moments of inertias were explicitly specified for the triangular element. Workstation simulations using this simple model are completed in a few minutes. The predicted nodal accelerations, velocities, and displacements at the gear attachment nodes are compared with the corresponding experimental data. These comparisons allow modifications to be quickly evaluated. Once adequate experimental/analytical correlation was achieved, the simple fuselage representation was replaced by a rigid fuselage model with the accurate geometry. The accurate geometry was needed to determine fuselage impact and to estimate the time to change from rigid to flexible simulation.

Crash Finite Element Model

The crash finite element model, see Figure 6, was developed from an existing modal vibration finite element model. Numerous modifications to the modal vibration model were required. These included: (1) reduction of the number of concentrated masses via mass distribution through material densities; (2) reduction of the number of different laminated composite shell and beam properties, where appropriate; (3) removal of very small elements, particularly in the tail, which are not relevant to the crash analysis; and (4) rediscretization to a finer mesh along the keel beams where significant crushing occurred. Additional details regarding the model conversion can be found in Reference [7].

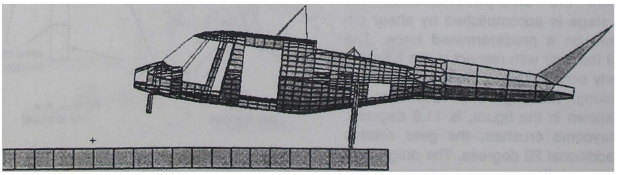


Figure 6. Crash finite element model.

The impact surface was modeled with 250 hexahedral solid elements with fixed bottom nodes. The impact surface elements were assigned material properties typical of steel.

The initial impact velocities were determined from photographic motion picture analysis to be - 462 in/sec vertical velocity, -384 in/sec horizontal velocity and no lateral velocity. Due to the pendulum motion of the test, an initial angular velocity of 9.6-degrees/sec pitch was induced just before left gear contact. These fuselage velocities were implemented in the finite element code using the rigid model with MATRMRG1 and MATRIG options in MSC.Dytran. The impact attitude angles of 6.25 degrees pitch up and 3.5 degrees roll left were incorporate by rotating the impact plane with respect to the fuselage.

The resulting flexible fuselage model was composed of 4,128 nodes and 7,346 elements (3,118 beam and rod elements, 3,283 quadrilateral shell elements, and 695 triangular shell elements). These elements were defined using 34 properties in the flexible fuselage model. The 98 concentrated masses represented actual lumped masses on the test article. For the rigid model simulations presented in this paper, all the materials were combined into a MATRIG except for the landing gear beams and tire masses. The fuselage mass, center-of-gravity and moments of inertia were explicitly specified in the rigid fuselage model. These values were derived using MSC.Nastran [8], where the element and mass information from the crash model was converted to MSC.Dytran format. The final fuselage model weighed 7,998 lb with a center-of-gravity at x=203.7 in., y=0.0 in., and z=87.0 in. The experimental test article weighed 7,832 lb with a center-of-gravity location at x-198 in. and z=100 in. The experimental determination of the z-coordinate of the cg-location is approximate due to the accuracy of the measurement set-up.

CORRELATION OF SIMULATION RESULTS WITH EXPERIMENTAL DATA

The rigid fuselage crash finite element model was executed in MSC.Dytran. The simulation results required nearly one hour CPU-time on a Sun Ultra 450 workstation to complete the 0.100-sec problem time. The accuracy of the simulation results was evaluated through correlation with the experimental sequence of events and the velocities at specific locations.

Figure 7 shows the predicted spring forces representing the landing gears, which were applied to the fuselage model. Note that large forces are initially experienced due to the high velocity differential. The force of the main landing gear is removed at the time of the fuselage impact, 0.095 sec. The large initial forces for the main gear are not physical. These large forces did not significantly affect the rigid fuselage response, presumably due to the very short duration.

The simulation sequence of events, see Table 1, was determined from the landing gear force curves. The test times were based on analysis of the high-speed film. Note that overall the correlation is very good with no more than four ms error. The longer time between the left and right gear contact for the simulation could be expected based on the simplified landing gear approach.

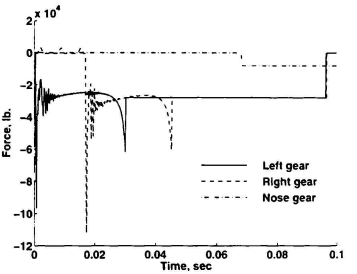


Figure 7. Predicted forces of the landing gear springs as a function of time.

Table 1. Timing of sequence of events for aircraft contact.

Event	Pred., sec	Test, sec	Diff., sec
Left gear	0.0	0.0	0
Right gear	0.016	0.012	-0.004
Nose gear	0.068	0.069	+0.001
Fuselage	0.095	0.0975	+0.0025

Although this paper focuses on the simulations prior to fuselage contact, the primary emphasis of the project was to accurately predict the input accelerations to the simulated occupants as well as retention of the overhead masses. Therefore, accuracy of the floor and large overhead mass accelerations was of utmost importance. Representative vertical accelerations at the pilot seat floor and right engine locations are shown in Figure 8. The measured acceleration data was filtered with a 3rd order Butterworth filter at 60 Hz. The data were filtered forward and backward to eliminate phase shifts resulting from the filtering process. The maximum acceleration values for the curves occur well after the fuselage impacts.

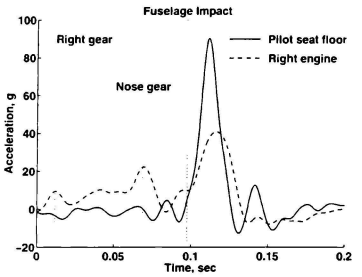


Figure 8. Representative experimental acceleration time histories for pilot seat floor and right engine locations.

For this paper, the initial velocity values at left gear contact were estimated based on the experimentally determined center-of-gravity translational and rotational velocities. The experimental velocity curves were computed by integrating the unfiltered experimental acceleration values as a function of time. Only the experimental and analytical "vertical" velocities would be compared in this paper. Note that the "vertical" velocity orientation for the simulations remained fixed with respect to the ground and was aligned with the aircraft at left gear contact. The experimental accelerometers were attached to the fuselage and thus rotated with the aircraft as the landing gear stroked. For this reason, the experimental and analytical "vertical" accelerations are aligned at 0 ms, but the acceleration directions would differ approximately 6.25° at fuselage impact. This discrepancy is not considered significant for the results presented here, since the cosine of 6.25° is 0.994, or the induced error is less than 1 %.

Comparisons of the two crew-seat floor attachment velocities are shown in Figures 9 and 10. Both the predicted and the measured velocities at the pilot and copilot locations show a nearly constant velocity prior to fuselage impact. This trend results from the balancing of the rotational pitch velocity component with the downward linear motion of the center-of-gravity.

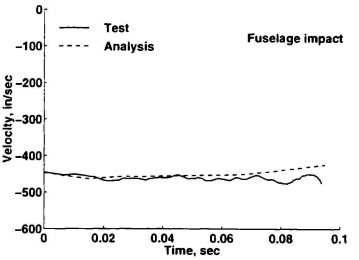


Figure 9. Comparison of experimental and analytical vertical velocities at the pilot seat floor.

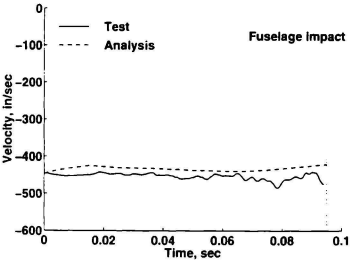


Figure 10. Comparison of experimental and analytical vertical velocities at the copilot seat floor.

The measured and predicted velocities near the troop-seat attachment locations are shown in Figures 11 through 13. Note that for these positions, the velocity decreases approximately 100 in/sec while the landing gear strokes. The measured and predicted vertical velocities above the troop seat area and at the rotor head are shown in Figures 14 and 15. The change in velocities is nearly 150 in/sec. The close comparison of the predicted and measured values, shown in Figures 9 through15, indicates that the simplified landing gear modeling implementation and the rigid fuselage assumption are adequate up to the time of fuselage impact.

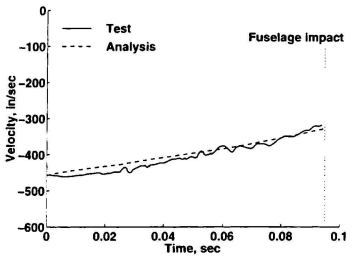


Figure 11. Comparison of experimental and analytical vertical velocities at the left troop seat floor location.

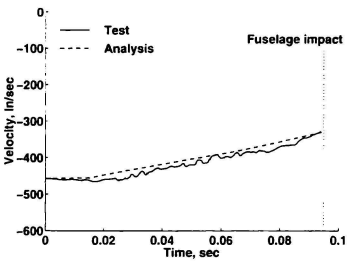


Figure 12. Comparison of experimental and analytical vertical velocities at the right troop seat floor location.

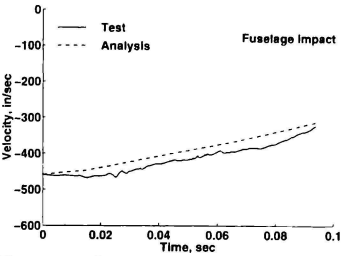


Figure 13. Comparison of experimental and analytical vertical velocities at BS182.

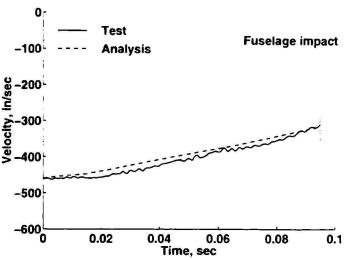


Figure 14. Comparison of experimental and analytical vertical velocities at BS188.

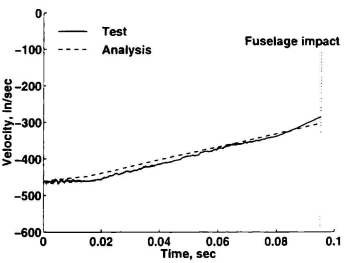


Figure 15. Comparison experimental and analytical vertical velocities at the rotor head.

In Figures 16 through 19, comparisons of the experimental and predicted velocity responses at locations aft of the center of gravity are shown. The right landing gear attachment point is shown in Figure 16. The simulation/experimental correlation at this location is fair. The irregular trend in the experimental data results from complexities of the landing gear stroking mechanism, such as shear pin failures and the fact that the test article is attached to a flexible fuselage rather than the rigid fuselage used in the simulations. These large oscillations in the velocities are local and not transmitted to the rest of the fuselage as indicated by the absence of the oscillations in the remaining velocity curves. The right and left engine velocities are shown in Figures 17 and 18, respectively. The correlation is excellent up to nearly 0.070 s. A similar trend is seen for the location at the top of the bulkhead at BS255, see Figure 19. A number of factors may contribute to the deviation of the simulation results from the experimental data, including violation of the rigid fuselage assumption in that the roof is beginning to collapse. In addition, the tail was determined to fail at 0.075 sec. This failure could have significantly affected the response at this portion of the fuselage.

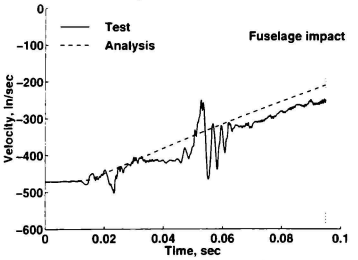


Figure 16. Comparison of experimental and analytical vertical velocities at the top of the right gear.

The accuracy of the acceleration response is directly related to the accuracy of the input energy at fuselage contact. One method to evaluate the accuracy of the impact energy is to correlate analytical and experimental squared velocities, see Figure 20. The squared velocity values are proportional to the kinetic energy. Prior to the correlation with the test data, the rigid fuselage approach would have been assumed valid up to the time of fuselage contact at 0.095 sec. However preliminary correlations of the experimental data and simulation results indicated that the flexible model should begin at 0.065 sec, or when the predicted velocities of the large masses aft of the

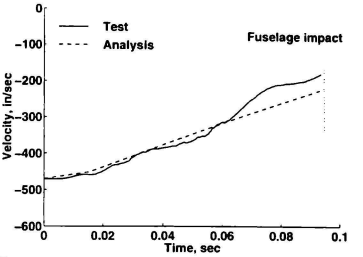


Figure 17. Comparison of experimental and analytical vertical velocities at the right engine.

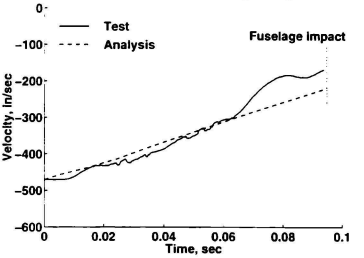


Figure 18. Comparison of experimental and vertical analytical velocities at the left engine.

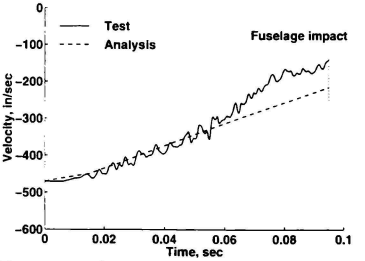


Figure 19. Comparison of experimental and analytical vertical velocities at the roof BS255.

center-of-gravity deviated from the experimental values. For these reasons, the accuracy of the rigid simulation results for utilization in a flexible finite element model has been evaluated based on a comparison of the experimental and analytical squared-velocity values at both 0.065 and 0.095 sec. The experimental value is used for the normalization.

At 0.065 sec, the predicted results at seven of the eleven locations are within 5 % of the experimental values. The accuracies of the predicted results are within 15 % at an additional three locations. The largest deviation is 23 % at the Roof BS255 location. These correlation values are considered excellent.

At 0.095 sec, the predicted squared-velocities at five of the eleven locations are within 15 % of the measured values. The predictions at an additional three locations are within 30 %. The predicted values at three of the locations vary substantially from the measured values. All of the floor locations, which would significantly affect occupant response, are within 30 %. The correlation between the experimental and analytical squared-velocities forward of the center-of-gravity is fair.

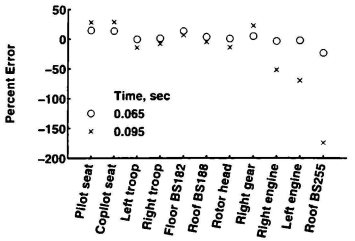


Figure 20. Error of the predicted squared-velocity at the indicated locations when compared to the experimental values (% error = $[(V^2_{test} - V^2_{pred})/V^2_{test}]x$ 100).

CONCLUDING REMARKS

An innovative and simplified landing gear modeling approach was implemented in a detailed crash simulation of a full-scale crash test of a composite helicopter. The crash simulation methodology was further simplified by utilizing the simple landing gear model with a rigid and geometrically accurate fuselage model. Helicopter crash simulations using this approach were compared with previously acquired experimental data from a full-scale crash test of a composite helicopter. The following conclusions were reached:

1) The simplified landing gear modeling approach accurately simulated the magnitude and orientation of the landing gear force on the fuselage.
2) The use of a rigid fuselage model for a portion of the crash simulation was appropriate and resulted in significant reduction of CPU-time for the total crash simulation.

In summary, the crash model was successful at approximating the effect of the landing gear stroking without adding substantially to the model complexity. This simplification in conjunction with the rigid fuselage assumption can result in significant reduction in CPU-time.

REFERENCES

1) Gamon, M.; Wittlin, G.; and LaBarge, B., "KRASH 85 User's Guide – Input/Output Format," Final Report DOT/FAA/CT-85/10, May 1985.
2) Anon, *LS-DYNA3D User's Manual*, Livermore Software Technology Company, Livermore, CA, 1997.
3) Anon, *MSC/DYTRAN User's Manual Version 4.0*, The MacNeal-Schwendler Corporation, Los Angeles, CA, November 1997.
4) PAM-CRASH, Engineering Systems International SA, F-94588 Rungis, France.
5) Anon., "ACAP Interim Technical Report, Volume III – Military Characteristics", Report No. TR17304, Sikorsky Aircraft, March 1982.
6) Boitnott, R. L., Jackson, K. E., and Fasanella, E. F., "Full-Scale Crash Test of the Sikorsky Advanced Composite Airframe Program Helicopter", Proceedings of the American Helicopter Society 56th Forum, Virginia Beach, VA, May 2-4, 2000.
7) Fasanella, E. L., Jackson, K. E., and Lyle, K. H., "Finite Element Simulation of a Full-Scale Crash test of a Composite Helicopter", Proceedings of the American Helicopter Society 56th Forum, Virginia Beach, VA, May 2-4, 2000.
8) Lahey, R. S., Miller, M. P., Reymond, M., "MSC/NASTRAN Reference Manual, Version 68," The MacNeal-Schwendler Corporation, Los Angeles, CA, 1994.

A00-37227

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

AIAA-2000-4091

STATE REPRESENTATION AND SIMULATION OF 2D TURBULENT WIND

N. Imbert*, Ph. Mouyon*, G. Montseny†
*ONERA/DCSD, B.P. 4025, F31055 Toulouse, France
†CNRS/LAAS, 7 av. du colonel Roche, F31077, Toulouse, France
e-mail: imbert@cert.fr, mouyon@cert.fr, montseny@laas.fr

Abstract: This paper presents the development of a Markovian time simulator for two-dimensional turbulent wind. This simulator generates wind components signals with not only specified time correlations but also specified spatial correlations along the wing. Wind spectra and cross-spectra may be of any type, even experimentally estimated, since rationality is not assumed. For that purpose we introduce a new tool for spectral factorization which is applicable to solve the state representation problem for general multidimensional non rational spectra. It is based on the use of a diffusive equation and on decoupling by spatial Fourier transformation. The obtained state model is linear but infinite dimensional. The discretization of this exact model leads easily to finite dimensional approximations over a prescribed frequency range. Stability is always guaranteed.

Keywords: Turbulent Wind. 2D stochastic process. Markovian simulation. Diffusive representation.

1 Introduction

Fine tuning of autopilots implies a good knowledge and precise simulation of turbulence effects on the plane. It is of importance to simulate a turbulence which shows the statistical characteristics of that met in flight. We present in this paper a methodology which allows to generate each component of the turbulent wind $z(y,t)$, with desired temporal correlations (t variable), and also space correlations (y variable) described by ideal models resulting from a subtle physical analysis of turbulence. The spanwise coupling effects are thus taken into account. It will be then possible for example to study the influence of a spatially not uniform turbulence on a wide-span wing, and more especially the mechanical loads undergone by a flexible wing.

The two main difficulties encountered to generate such signals are: the two-dimensionnal (2D) aspect and the non rationality of the involved spectra or cross-spectra.

As regards 1D turbulent wind, the theoretical spectral models of Dryden and Von Karman are well known. They are commonly used to simulate the three components of the gust field encountered by a plane. Dryden spectrum is rational with respect to $-s^2$ (where s is the Laplace variable). Thus a wind component with such a spectrum is simulated by filtering a white noise process b through a linear finite dimensional filter. The filter transfer function is easily found by factorization of the spectrum. Rational approximations of the Von Karman spectrum are also often used.

In section 3 we develop a general method for the factorization of any non rational spectrum. It is based on the introduction of an infinite dimension state space model (a diffusive equation). Finite dimensional approximations over a prescribed frequency range are then proposed, with arbitrarily chosen order (section 4).

The 2D simulation problem is dealt with in section 5. A spatial Fourier transform is used to decorrelate the signals $z(y,t)$ with respect to the spatial variable y. A collection of diffusive models allows to generate all uncorrelated involved signals. The spatial correlations are then restored by inverse Fourier transform. The model remains linear Markovian. The computation of finite dimension approximations of this exact solution relies on a discrete Fourier transform with respect to y, complementary to the use of approximated models for the generation of decorrelated signals. The numerical results presented in section 6 show the good adequacy of the approximated models with the theoretical 2D spectral model for the simulation of two-dimensional wind turbulence.

This work was supported by SPAé.

Copyright © 2000 by ONERA. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

2 Theoretical models

We recall in this section the analytical expressions of the spectra and cross-spectra[1,2] of the vertical component of the turbulent field. Other components have similar models, and are assumed to be independent from each other. These models depend on three parameters:

- σ the standard deviation of turbulence,
- L the scale of turbulence,
- V the longitudinal speed of the plane.

The cross-correlation function of the random signals $z(y,t)$ is defined by:

$$K(\Delta t, \Delta y) = E\left(z(y,t)\ z^*(y+\Delta y, t+\Delta t) \right)$$

where $E(.)$ denotes the stochastic mean. For the ideal Von Karman model (more precise than the Dryden model), an analytic expression of $K(\Delta t, \Delta y)$ is known (1). It involves modified Bessel functions of second kind K_q and of rational order q.

$$K(\Delta t, \Delta y) = \sigma^2 \frac{2^{\frac{2}{3}}}{\Gamma(\frac{1}{3})} \left[u^{\frac{1}{3}} K_{\frac{1}{3}}(u) - \frac{1}{2} u^{\frac{4}{3}} K_{\frac{2}{3}}(u) \right] \quad (1)$$

with $u = r/(aL)$, and where r is the distance between the two points considered:

$$r = \sqrt{\Delta x^2 + \Delta y^2} = \sqrt{V^2 \Delta t^2 + \Delta y^2}$$

The constant a is defined by

$$a = \Gamma(\frac{1}{3})/\Gamma(\frac{5}{6})/\sqrt{\pi} \approx 1.339$$

The frequential characterization of the signals z is obtained by Fourier transform $\mathcal{F}_t$ with respect to the time variable. We denote ω_t the pulsation associated with the time variable t. The transform of $K(\Delta t, 0)$ and $K(\Delta t, \Delta y)$ by $\mathcal{F}_t$ are respectively the spectrum (2) and cross-spectrum (3) of $z(y,t)$.

$$\begin{aligned} \widehat{K}_t(\omega_t, \Delta y) &= \mathcal{F}_t(K) \\ &= \int_{-\infty}^{+\infty} K(\Delta t, \Delta y)\, e^{-j\omega_t \Delta t}\, d\Delta t \end{aligned}$$

With $\omega = \omega_t / V$, the spectrum writes:

$$\widehat{K}_t(\omega_t, 0) = \sigma^2 \frac{L}{\pi} \frac{1 + \frac{8}{3}(aL\Omega)^2}{\left[1 + (aL\Omega)^2\right]^{\frac{11}{6}}} \quad (2)$$

And the cross-spectrum is given by:

$$\widehat{K}_t(\omega_t, \Delta y) = \sigma^2 \frac{2^{\frac{2}{3}}\, a\, L}{\Gamma(\frac{1}{3})\sqrt{2\pi}} \left[\frac{8}{3} w^{\frac{5}{6}} K_{\frac{5}{6}}(v) - w^{\frac{11}{6}} K_{\frac{11}{6}}(v) \right] \quad (3)$$

with

$$v = \frac{\Delta y}{aL} \sqrt{1 + (aL\Omega)^2} \quad \text{and} \quad w = \frac{\Delta y}{aL} \frac{1}{\sqrt{1 + (aL\Omega)^2}}$$

In order to simulate the random field $z(y,t)$ on any time period, it is necessary to have a Markovian representation of this field, i.e. a state equation which generates signals having the same cross-correlation, or in an equivalent way the same cross-spectrum. The construction of this state equation is uneasy in the present case because the spectrum is non rational and furthermore the signals are two-dimensional.

3 Diffusive representation of non rational spectra

This section solves the one-dimensional factorization problem. The problem is to find a Markovian representation of a 1D random field defined by its complexe spectrum $\Phi(s)$, where s is the Laplace variable. The spectrum is given by $\widehat{K}_t(\omega_t, 0) = \Phi(j\omega_t)$. It is possibly non rational. This result generalizes those known for finite dimensional systems within the standard framework of rational spectra[3,4].

For that purpose we seek the filter with transfer function $H(s)$ such as: $\Phi(s) = H(s)\, H(-s)$. The filtering of a white noise $b(t)$ (with spectrum $\Phi_{bb}(s) = 1$) by H, will then generate a signal $z(t)$ of spectrum $\Phi_{zz}(s) = \Phi(s)$. The first significant result of this article is described hereafter. It relies on the use of a diffusive equation[7] with random input, and whose parameters are determined by solving an infinite dimensional Riccati equation. First of all let us introduce some definitions.

Definition 3.1 *The operator $\nu(\xi)$ is defined by:*

$$\nu(\xi) = 2\,\xi\, \mathcal{L}^{-1}\left(\mathcal{L}_{II}^{-1}\left(\Phi(s) \right) \right)(\xi) \quad (4)$$

where $\mathcal{L}$ is the Laplace transform and $\mathcal{L}_{II}$ denotes the bilateral Laplace transform.

Definition 3.2 *The operator P is the hermitian operator solution of the nonstandard Riccati equation:*

$$AP + PA^* - (PA^*C^* + S)R^{-1}(CAP + S^*) + Q = 0 \quad (5)$$

with $R = \int_0^{+\infty} \nu(\xi)\, d\xi$, and where the other operators are defined by ($*$ stands for the dual operator):

$$C : x(\xi) \to \int_0^{+\infty} x(\xi)\, d\xi \qquad A : x(\xi) \to -\xi x(\xi)$$
$$S = \nu^* \qquad Q : x(\xi) \to \nu(\xi) x(\xi)$$

We can now state the result.

Theorem 3.3 *A Markovian representation of a 1D random process with spectrum $\Phi(s)$ is given by the infinite dimensional system:*

$$\left\{ \begin{array}{lcl} \dot{x}(\xi,t) & = & -\xi\, x(\xi,t) + B(\xi)\, b(t) \\ z(t) & = & \displaystyle\int_0^{+\infty} x(\xi,t)\, d\xi \end{array} \right. \tag{6}$$

where the operator B is defined by

$$B = (PA^*C^* + S)R^{-1} \tag{7}$$

It must be noted that generally the Riccati equation to be solved is not standard since Q is not definite nonnegative. This happens whenever the spectrum is not strictly decreasing with ω_t, because in that case the operator ν must not be positive.

However the existence of a solution to the Riccati equation is guaranteed by the constraint of positivity of the spectrum to be represented $\Phi(j\,\omega_t) \geq 0$.

4 Finite dimensional approximation

The previous section gives an exact theoretical solution to the 1D spectral representation problem in the non rational case. This solution is of infinite size. The development of a simulator thus requires the calculation of an approximate finite dimensional solution.

The proposed diffusive representation is particularly well fitted to this calculation because the discretization of the diffusive equations according to the ξ variable, is well known and perfectly mastered.

In fact the main difficulty we are faced to is the computation of ν. We propose hereafter a weak approximation of this operator whose computation only requires to solve a simple least mean square problem.

4.1 Computation of ν

Equation (4) is easily seen to be equivalent to:

$$\Phi(s) = \int_0^{+\infty} \frac{\nu(\xi)}{\xi^2 - s^2}\, d\xi \tag{8}$$

Indeed, according to (4) one has :

$$\begin{aligned} \Phi(s) &= \mathcal{L}_{II}(\,\mathcal{L}(\frac{\nu(\xi)}{2\xi})\,)(s) \\ &= \int_{\tau=-\infty}^{+\infty} \left[\int_{\xi=0}^{+\infty} \frac{\nu(\xi)}{2\xi} e^{-\xi\tau}\, d\xi \right] e^{-\tau s}\, d\tau \\ &= \int_{\xi=0}^{+\infty} \frac{\nu(\xi)}{2\xi} \left[\int_{\tau=-\infty}^{+\infty} e^{-s\tau} e^{-\tau\xi}\, d\tau \right] d\xi \\ &= \int_{\xi=0}^{+\infty} \frac{\nu(\xi)}{2\xi}\, \mathcal{L}_{II}(\,e^{-s\tau}\,)(\xi)\, d\xi \\ &= \int_{\xi=0}^{+\infty} \frac{\nu(\xi)}{2\xi} \frac{2\xi}{\xi^2 - s^2}\, d\xi \end{aligned}$$

The spectrum thus writes as a linear integration in ν. An approximation of this integral on a finite n-points ξ-grid is chosen with:

$$\Phi_{approx}(j\,\omega_t) = \sum_{k=1}^{n} \frac{\nu_k}{\xi_k^2 + \omega_t^2} \tag{9}$$

The set of $\{\xi_k\}$ is selected to cover the frequency range of interest $[\omega_{min}, \omega_{max}]$. On this range the ξ_k can be selected in geometric progression so that the approximation is the best possible one in a log-log representation of the spectrum. Equation

$$\Phi(j\,\omega_t) = \Phi_{approx}(j\,\omega_t) \tag{10}$$

is solved with respect to ν_k in a least mean square sense for a rather large set of ω selected over the range $[\omega_{min}, \omega_{max}]$. The estimated ν_k can then be regarded as weak estimation of the $\nu(\xi_k)$.

It is worth noting that to improve the precision of the approximation one can be brought to choose some of the ξ_k out of the frequency range of interest.

During the calculation the only constraint is that the positivity of the spectrum is preserved. If not, no solution will be found to the Riccati equation that must be solved at the following stage to factorize the approximated spectrum.

Finally it is quite obvious that in the case of a rational spectrum this procedure remains less precise than one which uses an identification of the modes. On the other hand the proposed procedure is systematic and avoids the supervision of stability problems which frequently appear with frequential identification methods.

4.2 Approximate representations

A spectral representation of Φ_{approx} is then easily obtained since we come down to the finite dimensional case. Such a representation may be written as:

$$\left\{ \begin{array}{lcl} \dot{x}_k(t) & = & -\xi_k\, x_k(t) + B_k\, b(t) \\ z(t) & = & \displaystyle\sum_{k=1}^{n} x_k(t) \end{array} \right. \tag{11}$$

where the B_k are the components of the vector B always defined by (7). But now all the operators involved in the Riccati equation express

themselves in a matrix form as follow:

$$A = diag\{\xi_1, \xi_2, \ldots, \xi_n\} \quad C = [1, 1, \ldots, 1]$$
$$S = [\nu_1, \nu_2, \ldots, \nu_n]^T \quad Q = diag\{\nu_1, \ldots, \nu_n\}$$
$$R = \sum_{k=1}^{n} \nu_k$$

The approximate spectral representation obtained really corresponds to a discretization in ξ of the exact spectral representation (6). The state equation is simply written on the ξ-grid and the integration is approximated by the Euler formula. The convergence of the Riccati equation solutions obtained under various discretization schemes, towards the exact infinite dimensional solution, has been studied for example in reference[6] .

5 2D modelisation

We now tackle the general problem of the non rational spectral representation for 2D signals. Our aim is to build a Markov process that generates the signals $z(y,t)$ where y is the space variable. The time and spatial correlations are supposed to be known and described by the cross-correlation function $K(\Delta t, \Delta y)$, or by the cross-spectrum $\widehat{K}(\omega_t, \Delta y)$.

5.1 Exact solution

One uses the spatial Fourier transformation $\mathcal{F}_y$ in order to decorrelate the signals with respect to y. Let us pose

$$\hat{z}(\omega_y, t) = \mathcal{F}_y(z) = \int_{-\infty}^{+\infty} z(y,t) e^{-j\omega_y y}\, dy$$

These transformed signals have the property to be uncorrelated with respect to ω_y. It is indeed well-known that one has:[8]

$$E(\hat{z}(\omega_y, t)\, \hat{z}^*(\omega_y + \Delta\omega_y, t + \Delta t)) = 2\pi\, \delta(\Delta\omega_y)\, \widehat{K}_y(\Delta t, \omega_y) \qquad (12)$$

where

$$\widehat{K}_y(\Delta t, \omega_y) = \mathcal{F}_y(K) = \int_{-\infty}^{+\infty} K(\Delta t, \Delta y)\, e^{-j\omega_y \Delta y}\, d\Delta y$$

Now let us consider the bi-spectrum defined by:

$$\Phi(\omega_t, \omega_y) = \mathcal{F}_t(\widehat{K}_y) = \int_{-\infty}^{+\infty} \widehat{K}_y(\Delta t, \omega_y)\, e^{-j\omega_t \Delta t}\, d\Delta t$$

For all fixed ω_y, a Markovian model can be defined by the approach described in the preceding section so that the spectrum of $\hat{z}(\omega_y, t)$ is equal to $\Phi(\omega_y, \omega_t)$. All the signals $\hat{z}(\omega_y, t)$ are independent with respect to ω_y if the white noises used to generate them are selected independent.

By inverse spatial Fourier transform one obtains then the correlated signals $z(y,t)$. We can now state the searched result.

Theorem 5.1 *A Markovian representation of a random process 2D with cross-spectrum $\widehat{K}_t(\omega_t, \Delta y)$ is given by the system of infinite dimension:*

$$\begin{cases} \dot{x}(\xi, \omega_y, t) = -\xi\, x(\xi, \omega_y, t) + B(\omega_y, \xi)\, b(\omega_y, t) \\ z(y,t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \int_{0}^{+\infty} x(\xi, \omega_y, t)\, e^{j\,\omega_y\, y}\, d\xi\, d\omega_y \end{cases} \qquad (13)$$

where the operator $B(\omega_y, .)$ is defined for each ω_y in a similar way than the operator $B(.)$ of theorem (3.3).

5.2 Real exact model

It may be noted that by construction the signals $\hat{z}(\omega_y, t)$ are real. It is generally not true for $z(y,t)$ obtained in the output equation by inverse Fourier transform of $\hat{z}(\omega_y, t)$. However, as by construction the $\hat{z}(\omega_y, t)$ are decorrelated, it is quite easy to build a real signal $\tilde{z}(y,t)$ with same cross-spectrum than $z(y,t)$. We will take:

$$\tilde{z}(y,t) = \Re e(z(y,t)) + \Im m(z(y,t)) \qquad (14)$$

In this case indeed one has:

$$\tilde{z}(y,t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \hat{z}(\omega_y, T)\, (\cos\omega_y y + \sin\omega_y y)\, d\omega_y$$

Because of the $\hat{z}(\omega_y, t)$ decorrelation property (12), the cross-correlation function of $\tilde{z}(y,t)$ writes as:

$$\begin{aligned} \widetilde{K}(\Delta t, \Delta y) &= E\left(\, \tilde{z}(y,t)\, \tilde{z}^*(y + \Delta y, t + \Delta t)\, \right) \\ &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} \widehat{K}_y(\omega_y, \Delta t)\, (\cos\omega_y y + \sin\omega_y y) \\ &\quad \times (\cos[\omega_y(y + \Delta y)] + \sin[\omega_y(y + \Delta y)])\, d\omega_y \end{aligned}$$

The term in factor of $\widehat{K}_y$ under the integral develops in: $(\cos\omega_y y + \sin\omega_y y)^2\ \cos\omega_y \Delta y\ +\ (\cos\omega_y y + \sin\omega_y y)\ (\cos\omega_y y - \sin\omega_y y)\ \sin\omega_y \Delta y$. Taking into account the fact that, for Δt fixed, $\widehat{K}_y(\omega_y, \Delta t)$ is a spectrum thus an even function of ω_y, the only non zero term after integration is:

$$\widetilde{K}(\Delta t, \Delta y) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \widehat{K}_y(\omega_y, \Delta t)\, \cos\omega_y \Delta t\, d\omega_y$$

Still thanks to the parity of $\widehat{K}_y$ in ω_y, one then recognize the inverse Fourier transform of $\widehat{K}_y$. Thus we actually have: $\widetilde{K}(\Delta t, \Delta y) = K(\Delta t, \Delta y)$. From where,

Theorem 5.2 *A real Markovian representation of a 2D random process with cross-spectrum $\hat{K}_t(\omega_t, \Delta y)$ is given by the infinite dimensional system:*

$$\begin{cases} \dot{x}(\xi, \omega_y, t) = -\xi\, x(\xi, \omega_y, t) + B(\omega_y, \xi)\, b(\omega_y, t) \\ z(y,t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \int_{0}^{+\infty} x(\xi, \omega_y, t)\, (cos\omega_y\, y \\ \qquad + sin\omega_y\, y)\, d\xi\, d\omega_y \end{cases} \tag{15}$$

where the operator $B(\omega_y, .)$ is defined for each ω_y in a way similar to the operator $B(.)$ of theorem (3.3).

5.3 Approximations

The computation of finite dimensional approximations of this exact solution relies on the use of a discrete spatial Fourier transform (computed by FFT). It involves a finite number q of spatial pulsations ω_{y_m}. There are thus two grids, one in ξ and the other in ω_y.

Because of use of the FFT, the ω_y-grid is actually fixed by the grid chosen for the space variable y. The y-step must be small enough with respect to the space correlation distance in order to limit the aliasing problems.

For each pulsation ω_{y_m}, one applies the procedure described in section (4). An approximation of the spectrum $\Phi(\omega_{y_m}, \omega_t)$ is computed by estimating the $\nu_k(\omega_{y_m})$ components. The corresponding approximate spectral representation is a q state equation with the following form:

$$\begin{cases} \dot{X}_m &= A_m X_m + N_m\, b_m(t) \\ \hat{z}(\omega_{y_m}, t) &= C_m X_m \end{cases}$$

In fact A_m and C_m are independent of m, and only N_m varies with m. The $\hat{z}(\omega_{y_m}, t)$ are independent from each other if the input white noises are selected independent. By concatenation of all these representations one obtains a Markovian model that may be used to generate $\hat{z}(t) = [\hat{z}(\omega_{y_1}, t), \ldots, \hat{z}(\omega_{y_q}, t)]^T$. This model has the following structure:

$$\begin{cases} \dot{X} &= A X + N b \\ \hat{z} &= C X \end{cases} \tag{16}$$

with

$$A = diag\,\{A_m\} \qquad N = diag\,\{N_m\}$$
$$C^T = [C_1^T, C_2^T, \ldots] \qquad b^T = [b_1, b_2, \ldots]$$

In order to generate the multidimensional signal $z(t) = [z(y_1, t), \ldots, z(y_q, t)]^T$ it is then sufficient to apply the inverse discrete spatial Fourier transform (sine and cosine inverse transforms). This operation is a simple matrix multiplication which thus only modifies the output equation. The expected correlations between the z-signals are restored by this multiplication.

6 Experimental results

The proposed technique was applied to the definition of a Markovian simulator of 2D signals whose spectral characteristics are described in section (2). The figures hereafter show the comparison between the exact Von Karman model (solid line) and its approximation (dotted line). The frequency range over which the approximation is made is materialized by the vertical lines. One used a spatial y-grid with 64 points, that is to say approximately one point every 0.5m spreaded over the wing.

We consider an unfavorable case for the plane behavior, with a turbulence scale equal to half the wingspan and a relatively low true air speed V. The resulting spectrum is significant within the studied frequency range (figure 1). We check that all the spectra (calculated at the various points of the y-grid) are identical. The cross-spectrum represented on figure (2) is that between the signals at the ends of the wing, therefore with the y-positions where the space correlation is the weakest.

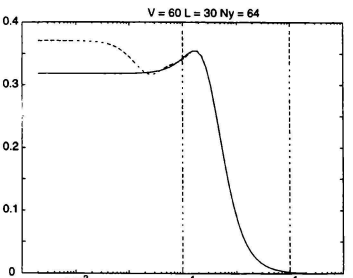


Fig. 1 Von Karman model - spectrum and approximations

The use of a logarithmic scale (figure 3) shows the non integer asymptotic behavior of the spectrum at high frequency.

This method can also be applied to experimentally estimated spectra. Indeed only the numerical values of the bi-spectrum (or the spectrum for 1D signals) at the points of the frequency grid are necessary. In the case of the turbulent wind recorded in flight, this approach allows, via a spectral estimation carried out for example with the tradi-

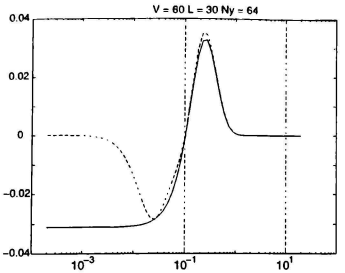


Fig. 2 Von Karman model - A cross-spectrum and its approximation

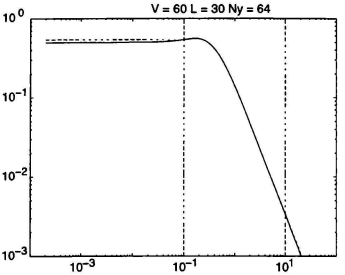


Fig. 3 Von Karman model - spectrum and approximations (log-log scale)

tional Welch technique, to simulate other flights under statistically equivalent conditions from the turbulence point of view.

References

[1]Ph. COSTES, Modélisation de la turbulence stochastique - première partie: Modèle de turbulence au centre de gravité, Rapport technique 512.0142/98, Aérospatiale Aéronautique, (1998).

[2]J. HOUBOLT, A. SEN, Cross-spectral functions based on Von Karman's spectral equation, Technical Report NASA CR-2011, Aeronautical Research Associates of Princeton, (1972).

[3]J. LEWIS, Optimal estimation, Wiley, (1986).

[4]K. ZHOU, J. C. DOYLE, K. GLOVER, Robust and optimal control, Prentice Hall, (1996).

[5]SODESTROM, System Identification, Prentice Hall, (1989).

[6]H. T. BANKS, R. C. SMITH, Y. WANG, Smart material structures. Modeling, estimation and control, Research in Applied Mathematics, (1996).

[7]G. MONTSENY, Diffusive representation of pseudo-differential time-operators, ESAIM: http://www.emath.-fr/proc/vol.5/, p. 159-175, (1998).

[8]M. LABARRERE, J. P. KRIEF, B. GIMONET, Le filtrage et ses applications Cépaduès Eds, Toulouse, (1988).

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37228

AIAA-2000-4093

PREPARING METEOROLOGICAL DATA FOR SIMULATIONS USING SEDRIS

Louis Hembree
Naval Research Laboratories
Marine Meteorology Division
7 Grace Hopper Ave
Monterey, CA 93943
831-656-4787
hembree@nrlmry.navy.mil

Rob Cox
SAIC
12479 Research Parkway Suite 600
Orlando, FL 32826
407-207-3609
ROBERT.M.COX@cpmx.saic.com

Valerie Pastor
SAIC
550 Camino El Estero Suite 205
Monterey, CA 93940
831-649-5242
valerie.l.pastor@cpmx.saic.com

Keywords:
SEDRIS, EDCS, Atmosphere, GRIB Conversion, SEDRIS transmittal, visualization, VIS5D

ABSTRACT: The Synthetic Environment Data Representation and Interchange Specification (SEDRIS) project was developed to facilitate the representation, transfer, and reuse of environmental data in Modeling and Simulation (M&S) programs. This objective coincides with the Department of Defense (DoD) objective to analyze its fielded and prototype systems, train its warfighters, and simulate warfighting conditions using M&S. Without an accurate representation of the meteorological conditions that the warfighters may encounter, the DoD objective would not be fully met.

Therefore, meteorological data has been mapped into the SEDRIS. SEDRIS uses a hierarchical object-based structure that provides the user high flexibility in how data is represented and subsequently transmitted. It uses a standard data representation model and interchange mechanism and fully represents the environment by capturing all data elements and their relationships. SEDRIS also provides a standard data interchange mechanism and format to support the distribution of environmental data and promotes the sharing of databases between simulations.

During 1999, several activities have taken place to ensure that the M&S community can accurately represent and use meteorological data via SEDRIS. Initially, a mapping of DoD requirements to the SEDRIS Data Representation Model (SDRM) had to be accomplished. After the mapping was complete, an initial conversion program was established for Gridded Binary (GRIB) data. To assist users of SEDRIS meteorological data transmittals, an application was developed to view these data. That application will be presented and a discussion of how it allows the user of the SEDRIS to assure a lossless transmittal of the data will be presented. Other efforts were performed to expand the meteorological capabilities of SEDRIS to additional formats. The first non-GRIB format investigated was for meteorological observations. For this work, the SEDRIS 2.5.3 release was used. In June 2000, SEDRIS 3.0 was released and efforts are being made to extend the software to that version.

Release C: "This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States."

The Evolution SEDRIS

It has been the objective of the U.S. Department of Defense (DOD) for years to analyze its fielded and prototype systems' capabilities. To accomplish a portion of this objective, a modeling and simulation (M&S) activity was developed. A significant component in this M&S activity was the creation of databases that represent the physical world. These systems are faced with the challenge of establishing consistent and correlated databases at the lowest possible cost, both in terms of time and money.

The DOD attempted to address this problem in the 1980's with a multi-service program called Project 2851. The result of this project was the Standard Simulator Data Base (SSDB) Interchange Format (SIF) and the Standard Simulator DataBase Facility (SDBF) at Kirtland Air Force Base, New Mexico. The SIF mainly supports the distribution of terrain datasets. About the same time, the Simulation Networking (SIMNET) program began at the Defense Advanced Research Projects Agency (DARPA). The key concept was the networking of homogeneous and later heterogeneous simulations. These requirements highlighted the new environmental data representation needs. Further requirements were placed on the M&S community; in particular, those of the computer generated forces, high-density terrain, ocean, atmosphere, and space databases.

Without an effective way to interchange the full range of environmental data in the networked M&S community, it became apparent a new database interchange mechanism was required. Thus, the Synthetic Environment Data Representation and Interchange Specification (SEDRIS) program was initiated.

The SEDRIS vision

Data interchange is not just for reuse in building M&S databases, but is key for achieving interoperability between distributed, heterogeneous training systems/networks. The successful interchange of environmental data requires both a loss-less and an unambiguous transfer of data from one system to another. Loss-less in the sense that no data is lost during the conversion interchange process. Unambiguous in the sense that the meaning of all data interchanged is fully understood.

The SEDRIS vision is the creation of a data representation model that supports easy access to interchanged data through the use of Application Programmer Interfaces (APIs). The "data model" enables the open interchange of data by providing a common representation from which native (possibly proprietary) data can be converted to and from SEDRIS.

The SEDRIS objectives

The U. S. Defense Modeling and Simulation Office (DMSO) stated in the DOD Modeling and Simulation Master Plan that providing timely and authoritative representations of the environment is a core requirement to achieve interoperability among aggregated heterogeneous simulation systems. SEDRIS will be key to satisfying this requirement.

The specific objectives of SEDRIS are:

- By using a data representation model, capture the complete set of environmental data elements and their relationships.
- Implement a standard API for accessing data elements.
- Minimize the cost to access and reuse data by lowering the software barrier to entry.
- Provide a standard interchange mechanism between database builders and consumers.
- Facilitate interoperability of networked heterogeneous simulations.
- Support reuse of environmental databases between disparate simulations.
- Use the same data model as an access mechanism to import and export source data into and out of various database generation systems.
- Promote consensus understanding of the diverse requirements and implementation choices used within the M&S community through education

Current Core SEDRIS Software

For this work, the SEDRIS 2.5.3 release was used. It can be found at the SEDRIS web site — www.sedris.org. SEDRIS 3.0 was released in June 2000. Efforts are being made to extend the work accomplished here to the most recent version of SEDRIS.

SEDRIS Data Representation Model

The 2.5 release of SEDRIS includes three hundred and sixty SEDRIS classes. The OMT diagram covers twenty-three pages (and is available in either PostScript or PDF format). Figure 1 is a compressed DRM showing the main classes that could be used in the representation of atmospheric data. The data dictionary, which defines and describes each of the classes, is also

available in either plain text or in HTML. The easiest way to use the data model is to examine the OMT diagram while sitting next to an HTML browser with the HTML version of the data dictionary loaded. Within the HTML dictionary, all of the classes have the appropriate links to any and all related classes. For a single class, the dictionary describes the class, gives examples of how to use the class, has a list of Frequently Asked Questions (FAQs) with their answers for that class, and lists the ANSI C structures used by SEDRIS to represent the data for that class. With SEDRIS 3.0, SEDRIS transitioned to UML to depict its DRM.

Certain parts of the data model are complicated enough to be worthy of their own separate, more detailed explanation. Technical guides are being written to cover the following topics within the SEDRIS data model: Topology, Control Links, Data Tables, Hierarchical Index Tables, Images and Color Models, Attribute Inheritance and Contexts, and the Environmental Data Coding Standard (EDCS).

From all of these topics, only the EDCS will be discussed here and even then only briefly. The EDCS fulfills a very important need in SEDRIS — the need to be able to identify (label) an object. The EDCS defines a set of Environmental Classification Codes (ECC), Environmental Attributes Codes (EAC), and Environmental State Codes (ESC). All are important, but the ECC may be the most important. How does a user find what they are looking for inside of a SEDRIS transmittal? By knowing which ECC values to search for. Want to find railroads? Then look for objects in SEDRIS that have the code for railroad attached. Want to find waterfalls? Look for waterfalls based on the ECC value for waterfall. The same is true for atmospheric parameters and understanding the ECC or EAC identifier for the given parameter.

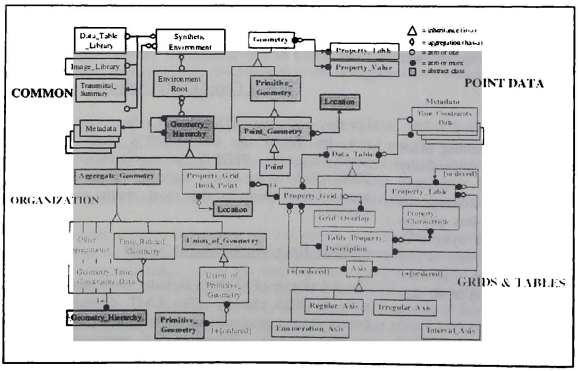


Figure 1: Compressed SEDRIS Data Representation Model (DRM) showing those classes most applicable for representing atmosphere data.

SEDRIS API

The SEDRIS API has been broken into distinct sections or 'functional groups.' ANSI C 'header' files define all SEDRIS APIs.

The Conversions API provides the 'low-level' functionality to convert coordinate values between coordinate systems, to convert color values between color models, and to convert measurements for certain units. The Conversions API is independent of all of the other SEDRIS APIs. This is a very important part of the SEDRIS core activities and the reader is directed to the SEDRIS web site for detailed discussions and papers.

The DRM API provides information about the SEDRIS data model. It is independent of any transmittal. It is primarily useful for developing applications that benefit from knowing the structure of SEDRIS classes. This structural information includes the text names of the classes, how many fields the classes have, the text names of the fields, the C types of the fields, the text names of the enumerated values, etc.

The Level 0 Read API is used to find and extract information from one or more SEDRIS transmittals. More than one transmittal can be open at the same time. There can be many implementations of the Level 0 Read API. It can be implemented "on" a native database or file so that it appears to a user as a SEDRIS transmittal, or it can be implemented to support the SEDRIS Transmittal Format (STF). The SEDRIS project has provided one such implementation that allows a user to read SEDRIS transmittals from STF files. SEDRIS documentation

at the web site fully explains the details on how to read SEDRIS data using the Level 0 Read API.

Two different approaches can be used to create a STF. The simplest, most direct method is to use the SEDRIS Write API. The other approach is to use the SEDRIS Traverse application. This latter approach can be used when the SEDRIS Level 0 Read API has been implementation on a native database or file format as mentioned above. Using the Write API to create the transmittal is the most efficient approach and will produce the more efficient transmittal. The STF is the standard file format for SEDRIS transmittals. STF is not a library or repository of databases, just as SEDRIS is not a library or repository of databases.

Various 'pass through' APIs also exist for the Level 0 Read API. The Debug API is an example. It acts as a thin layer between the user and whatever other API implementation is actually going to read the transmittal. The Debug API never reads a transmittal on its own. The Debug API does however keep track of objects to ensure that all objects for a transmittal are freed before the transmittal is closed, that only valid objects are used in data retrieval calls, etc. The Debug API is intended for use by SEDRIS consumers to help debug their applications.

SEDRIS Applications

There are several SEDRIS applications that have been developed. These applications allow a SEDRIS user to review and view the data for "correctness." It is left to the reader to visit the SEDRIS web site for a detailed discussion of the multiple SEDRIS applications.

Atmospheric Environment in SEDRIS

The actual incorporation of atmospheric parameters into the SEDRIS involved a process of mapping the requirements as stated by the DMSO Air and Space Natural Environment (ASNE) Modeling and Simulation Executive Agent (MSEA) to the existing EDCS and ensuring the EDCS was updated for any parameters that were required. The primary document used to map atmospheric requirements to SEDRIS is the ASNE Modeling and Simulation Baseline Requirements Assessment (ASNEM&SBRA). This report describes the ASNE requirements of a selected subset of major simulation programs that are now in development (e.g. JWARS, JSIMS JMASS, and STOW).

For the purpose of mapping the ASNE requirements into SEDRIS, we considered the requirements as listed by table 5-1 in the ASNEM&SBRA. Those requirements were listed by category of natural environment information (data, products, and tools) that is required by the M&S community. The table also cross-references the source documents that specifically state ASNE requirements for that information.

Based on the requirements table we determined whether the parameter was already in the EDCS and if it was we listed its EDCS code. Then we checked its current definition for accuracy and listed either yes or no if the definition was correct. The last column was a mapping to the Defense Data Dictionary System (DDDS) number for that atmospheric requirement. Table 1 is an extraction from the ASNE weather requirements table.

After all the parameters were mapped, then the process of developing the atmospheric converters for SEDRIS was begun. It was decided that the first converter would be the for GRIB data.

Atmospheric Forecast/GRIB Mapping

The mapping of atmospheric model forecast data to SEDRIS using the SDRM and the EDCS is discussed. Numerical atmospheric models are used to produce forecast data and simulated atmospheric environments. Atmospheric model output typically consists of fields at mean sea level, the earth's surface and at various levels in the atmosphere. The fields are also output for the base time (analysis time) and at forecast times usually expressed as a delta (tau) from the base time.

Table 1: Example weather requirements with associated EDCS and DDDS cross-references.

Parameter	In EDCS	EDCS Code	Correct Definition	DDDS Number
Temperature				
Surface	Y	ATS_	Y	
Freezing Level	N			
Inversion Height	Y	TILB	Y	36882
Tropopause Height	Y	TRH_	Y	26469
Pressure				
Altimeter Setting	Y	Pas_	Y	29637, 36848
D Value	N			
Pressure Altitude	N			
Density Altitude	N			
Sea Level Pressure	Y	PRMS	N	29634
Station Pressure	Y	PRE_	Y	29635,36415
Wind				
Gusts	Y	WGS_	Y	29413 (OBS), 36472 (FCST)
Gust Spreads	Y	WGR_	Y	45162

These fields are usually distributed as grids using the World Meteorological Organization (WMO) Gridded Binary (GRIB) format in the form of GRIB messages. These messages may each be a separate file or be concatenated together into one or several files. There is no implied organization to the GRIB messages except what might be provided by the data provider in a file naming convention and how the messages are concatenated or otherwise organized using some directory structure. The GRIB messages are in a simple flat structure, which is captured by SEDRIS.

Detail Mapping

This section presents the details about various aspects of the mapping of atmospheric parameters into SEDRIS. Figure 2 is a high level mapping for an atmospheric forecast data set as described above. The mapping includes the required classes Synthetic_Environment, Transmittal_Summary, Environmental_Root, Spatial_Domain, and metadata classes. Immediateky below these classes are nested Time_Related_Geometry classes, which serve to organize the data by base time and forecast tau. Below the Time_Related_Geometry class corresponding to the forecast taus, are the Property_Grid_Hook_Point classes, which serve to locate the grids with respect to the earth's surface. Multiple Property_Grid classes can be associated with each Propery_Grid_Hook_Point and define characteristics of each grid. Each Propert_Grid class can have multiple Table_Property_Description classes, which describe what is contained in each cell of the grid.

The Synthetic_Environment contains SEDRIS version information. The Transmittal_Summary class contains a summary of what is contained in the transmittal, classes, overall coordinate system, and hierarchies (Table 2). A consumer can use this information to decide if the transmittal contains what they need and if they can process the transmittal.

The Spatial Domain under the Environmental Root is used to locate the general area covered by the data set. It contains the coordinates of opposite corners of a box defined in the transmittal coordinate system. Two Geometry_Hierarchies are used in creating three levels of hierarchy: Time_Related_Geometry and Property_Grid_Hook_Point. Two Time_Related_Geometries are used; one to identify the base forecast time and one for the forecast deltas. These are indicated by the Classification_Data class. The fields of the Time_Related_Geometry provide information about the uniqueness and organization of the classes below the Time_Related_Geometry. Each child of a Time_Related_Geometry is identified using the link class Geometry_Time_Constraints_Data.

The first hierarchy, the first Time_Related_Geometry below the Environmental Root, corresponds to the base forecast time as identified by the Classification_Data class. The Geometry_Time_Constraints_Data link class for each child, points to an Absolute_Time_Point class whose fields specify the corresponding date and time. Each of these children contains a second hierarchy level, another Time_Related_Geometry, which corresponds to the forecast deltas (taus). In this case the Geometry_Time_Constraints_Data link class points to a Relative_Time_Point class which specifies the delta from the base forecast time as indicated by their association with the corresponding Absolute_Time_Point. Note that the Absolute_Time_Point can be shared between the Relative_Time_Points under the same hierarchy. This conserves space. Each of these children contains the

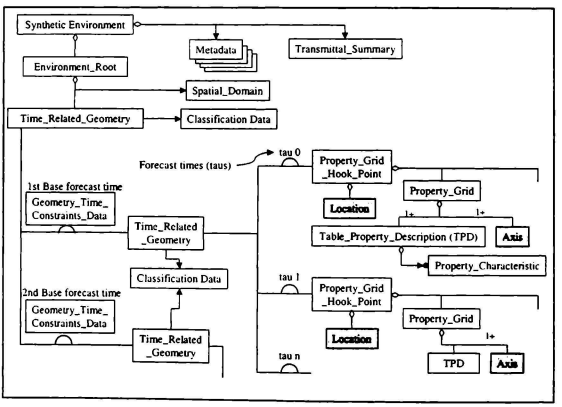


Figure 2: High level mapping for an atmospheric forecast data set.

Table 2: Contents of Transmittal_Summary class.

Transmittal_Summary
features_present = SE_NOT_PRESENT feature_topology_present = SE_NOT_PRESENT geometry_present = SE_PRESENT_IN_ENVIRONMENT_ROOT geometry_topology_present = SE_NOT_PRESENT data_tables_present = SE_PRESENT_IN_ENVIRONMENT_ROOT priority_values_present = SE_NOT_PRESENT mobility_values_present = SE_NOT_PRESENT thermal_values_present = SE_NOT_PRESENT terrain_lods_present = SE_NOT_PRESENT two_D_features_flag SE_NOT_PRESENT models_present = SE_FALSE images_present = SE_FALSE sounds_present = SE_FALSE symbols_present = SE_FALSE color_model = RGB SDCS_usage_list_is_comprehensive = SE_FALSE

third hierarchy level corresponding to the Property_Grid_Hook_Point. Note that while the Property_Grid_Hook_Points for each tau are probably identical in contents, the objects cannot be shared because of how inheritance in an object based system works. If two times shared the same Proprety_Grid_Hook_Point object, there would be not way to tell which time the Property_Grids under the Proprety_Grid_Hook_Point were associated with. However, the GDC_Location_2D object, which locates the Property_Grid_Hook_Point, can be shared.

Table 3 shows more detail for Property_Grids. One of two Property_Grids are shown, an analysis volume (the other is an analysis at the earth's surface). The Property_Grid field data_table_type uses a SEDRIS Classification Code to indicate the type of Property_Grid. The spatial_axes_count field gives the number of spatial axes. Notice that an atmospheric volume only has two spatial axes. This is because the vertical coordinates are not in units of height above the vertical datum. The normal atmospheric vertical units of pressure or sigma are not considered spatial units because they are not referenced to a fixed datum. Therefore, the vertical axis is not a spatial axis. The location_index identifies the grid point that corresponds to the Property_Grid_Hook_Point location. The coordinate system is defined as a Geodetic Coordinate System. The data_present field is a flag used to indicate if data are actually present. In some applications where the terrain is represented using Triangulated Irregular Networks (TIN), it is desired to be able to describe the grid from which the TIN was derived. In this case no data is actually contained in the property grid.

Table 3: Contents of Property_Grid class.

Property_Grid
data_table_type = SE_SCC_ATMOSPHERIC_ANALYSIS_VOLUME spatial_axes_count = 2 location_index = (0,0,0) coord_system_parms.coord_system = SE_GDC_S coord_system_parms.gdc_parameters.horizontal_datum =SE_SPHERICAL_COA_HDATUM coord_system_parms.gdc_parameters.vertival_datum = SE_SPHERICAL_COA_VDATUM coord_system_parms.gdc_parameters.elevation_units = SE_METERS data_present = SE_TRUE

Table 4 shows the axes classes referenced in Table 3. Note that the axes are ordered and that the spatial axes always come first. The two Regular_Axis classes define the two spatial axes. The class fields define the axes. Note that these two classes can be shared by the property grids. The vertical axis of the atmospheric volume is defined using the Irregular_Axis class. The actual values of the axis tick marks are supplied in an array.

Table 4. Axis Classes

Regular_Axis
axis_type = "SE_SAC_SPATIAL_GEODETIC_LATITUDE" axis_unit = "SE_UNITS_DEGREES_ARC" axis_value_count = 231 interpolation_type = SE_NOT_SUPPLIED" first_value = 29. spacing = .20 values_are_ints = SE_FALSE type_of_spacing = SE_LINEAR_SPACING axis_alignment = SE_ALIGN_LOWER

Table 5 shows example Table_Property_Description classes. A Table_Property_Description class is required for each parameter contained in the Property_Grid. The Table_Property_Description fields define the parameter, its units, and its value type. Each Table_Property_Description can have several Property_Characteristics classes associated with it. The Property_Characteristics class is used to provide additional information about the parameter. This class can be used to specify upper and lower bounds, significance digits, tolerance, missing data flags, and several other characteristics. The SEDRIS Write API will use these three characteristics when they are

Table 5: Table_Property_Description class contents.

Table_Property_Description
attribute_code = SE_SAC_WIND_SPEED_U_COMPONENT value_unit = SE_UNITS_METERS_PER_SECOND value_type = SE_DT_FLOAT_32

present to compress the data when it is written to the SEDRIS transmittal in the next release.

So far there has been no mention of the actual data only the description of how the data is organized. To this point there has been no discussion of where the data is put. This is because the data is not explicitly contained in the DRM. However, it is implicitly contained. Early on it was decided not to include the Data_Table and Property_Grid data in the DRM because the amount of data can be large. Instead, hidden pointers are maintained that point to the data. In a STF, the data is actually contained in files external to the DRM portion of the STF. When creating a STF, after the classes as described are created and added to a transmittal, the user makes additional function calls to add the actual data to the transmittal. When reading a STF, the opposite is done. Once a user has found the Property_Grid that they want, the user makes explicit calls to load the Property_Grid data.

GRIB to SEDRIS STF Conversion

Atmospheric forecast model output is usually distributed using the World Meteorological Organization (WMO) Gridded Binary (GRIB) format. A process was developed to convert GRIB files to a SEDRIS Transmittal Format (STF) using the above mapping. The process uses two programs, one to perform the mapping and the other to create the STF. The first program, GRIB2SED, analyzes the GRIB files and creates the mapping to SEDRIS. It creates a mapping file that describes how the GRIB data is to be mapped using the SEDRIS DRM. In addition to the GRIB files this program also requires a metadata record either in the form of a Federal Geographic Data Committee's Content Standard for Digital Geospatial Metadata Version 2 1998 metadata record or a file that uses the same tags to identify the metadata sections. The metadata record is required to provide additional information required in a STF that is not available in the GRIB records. After the mapping file has been created, the second program, GRIB2STF, reads the GRIB files and creates the STF.

This mapping file approach was taken because, while the general form of the mapping is the same from data set to data set, the actual mapping depends on the number of base forecast time, number of taus, the parameters, and the surface types. Also, by decoupling the mapping from the actual creation of the STF, future support for other formats is easier to provide.

Conversions have been done for COAMPS, MM5, and Wave Model (WAM) data sets. Execution time for a file of 510 GRIB messages took approximately 114 seconds. There is an increase of total data set size by a factor of 3. This is because the GRIB files use internal compression, which SEDRIS 2.5.3 does not use. Compression for Data_Tables and Property_Grids is in SEDRIS 3.0 release (June 2000).

To use the conversion programs, a user must provide three files: 1) a file that contains the GRIB table that contains the GRIB parameter information for their data set, 2) a file that maps the GRIB parameter IDs to SEDRIS Attribute Codes (SAC), and 3) a file that maps the GRIB level types to the type used within SEDRIS. These files are in a specific format. However, the existing files can be used with few if any changes for most GRIB data sets. Any changes required will reflect local additional and modification to the GRIB standard.

Viewing Atmospheric Data

There are many ways to ensure that the data that is converted to the STF is "correct." Within SEDRIS, there are many tools that a user can call on to verify that the conversion process worked as planned. For the atmospheric conversion process, a SEDRIS tool was developed to allow the user to view the resulting atmospheric files. This tool uses the freeware developed by the University of Wisconsin called VIS5D. VIS5D is a software package that displays standard meteorological parameters. The initial implementation creates a VIS5D file from a STF. This file is then viewed using VIS5D.

VIS5D requires a specific format for its input file. To create this file, the stf_to_vis5d program, reads the STF to determine the projection, the size of the grid, the parameters, and the times to be plotted. It then reads the data and creates the VIS5D input file. This is a multiple pass process, as the program needs to determine the amount of space required to store the information and because the structure of the VIS5D input file requires information that can only be obtained by performing a prescan of the STF. The data can then be read. In determining the times, duplicate times are eliminated by using only the most recent analysis and forecast times.

The time information is obtained from the Time_Related_Geometry and the associated Geometry_Time_Constrant_Data link objects. The projection information is retrieved using the Property_Grid_Hook_Point and Property_Grid objects. The parameters are determined form the Table_Propery_Description objects. Al of this can be determined without actually reading the data because the data is not actually stored in the Property_Grid object. The Property_Grid object contains a hidden pointer the SEDRIS READ API uses to retrieve the data when appropriate function call is made.

Figure 3 is an example of a VIS5D plot. The plot is of 500 mb vorticity (gray scale) and accumulated precipitation (contours). The original is in color and the conversion to gray scale looses information as different colors map to the same gray value.

Conversion of Observational Data to SEDRIS

There are several observational data format types that could be converted into STF. It was decided that the DATSAV3 format used at the Air Force Weather Agency would be the first.

The current DATSAV3 contains either surface observation files or upper air files. Initially, the main emphasis has been to develop two separate decoders for DATSAV3 (surface and upper air). It was decided that two separate decoders would be the best because the data definitions are so different. This way it minimizes the potential problems in creating the transmittal, thus it would be simpler to code, document, and be much easier to modify.

Each of the two data files (surface and upper-air) have a

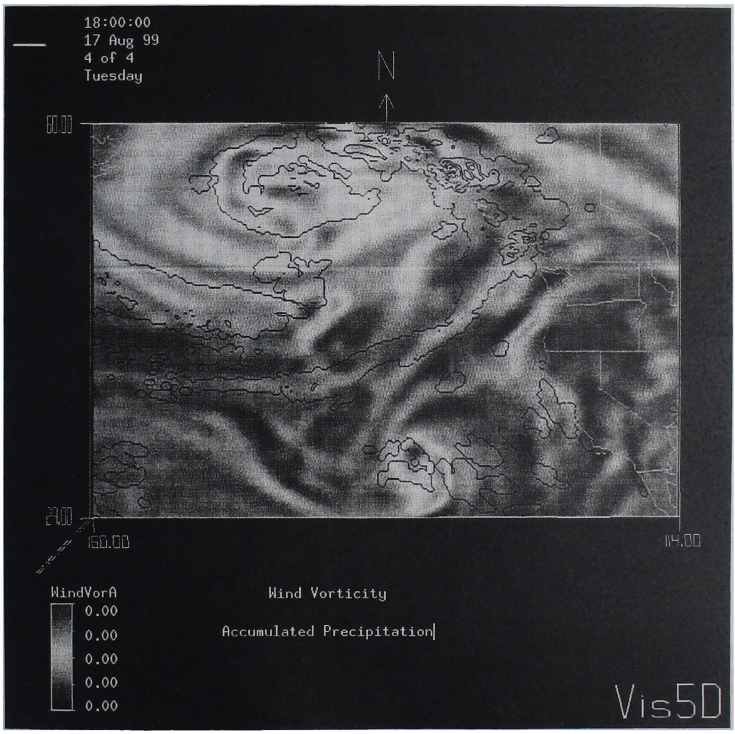


Figure 3: Example of VIS5D plot of 500 mb vorticity (gray scale) and accumulated precipitation

control data section, a mandatory data section, an additional data section, a remarks section, and an element quality data section. The first two sections are fixed in format.

The additional data section has multiple data parameters that can all be in the file, just a portion in the file, or no parameters are in the file. Some possible parameters for a surface observation are: liquid precipitation, snow depth, hail, and present weather observation.

The remarks section and the element quality section are also optional in their being in a particular data file. A remark is variable in length and runs from 0 to 250 characters long.

In the decoder, the fixed length data is read in. This data is placed into a structure. The file is then checked to see if it is at end of file. If so, the program exits. If not, the program reads a three-character type identifier and goes into a loop until an end of file is reached. Each data type and data section has a three-letter code, which is tested against the type identifier. For each data type, the additional data for that specific type is read and placed into a structure. As some data types have multiple reports, looping is done to read each report. When the file has been processed the procedure exits.

At this point the data is ready to be placed in the STF. The GRID-to-SEDRIS converter is the basis for that conversion. By developing a program to analyze the data files to create a mapping file, the same program that is currently used to create an STF from GRIB files can me used to create STFs from DATSAV3 or other format with minimal modifications. The modifications would consist of adding support for the classes required to represent observation type data and to read the additional format types. All map projections are considered and the different types of observational data (currently surface and upper-air), will be considered. The objective is to allow this STF converter to handle the multiple types of meteorological observation types. As stated, it converts surface and upper-air data. The architecture is such that as the requirements change and different data types are desired to be converted, then they can be added.

Conclusion

In this paper, we presented the work to convert atmospheric data in the SEDRIS Transmittal Format (STF). Initially, all requirements were mapped into the Environmental Data Coding Standard. Those requirements that were not in the EDCS were added. From there, the data were "mapped" into the SEDRIS. This mapping provides the users a structured analysis of how the atmospheric data is converted into the STF. It also provides the user a way or the knowledge of how to extract the data from STF. Once the data were in STF, a method of viewing the data was created using the freeware called VIS5D. This viewer enables the user a way to ensure that all data were properly converted and a way to analyze the data for inconsistencies. First, gridded atmospheric data were converted into the STF. Then observational data were formatted into the STF. Because there are many types of observational data, the Air Force Weather Agency's DATSAV3 format was chosen as the test case. In particular, the surface and upper-air observational data were converted. Thus, with the SEDRIS 3.0 release, gridded weather files are in STF and the observational data is being prepared for that formatting.

Author Biographies

Dr. Louis Hembree is a Research Meteorologist at the Naval research Laboratory Marine Meteorology Division, Monterey, CA. He has represented the Air and Space Natural Environment in the SEDRIS development as the subject matter expert for the atmosphere. He is also Technical Lead and Resource Site Lead for the Master Environmental Library.

Dr. Rob Cox is a Chief Scientist with SAIC. Dr. Cox serves as project director and principal investigator for SEDRIS. His Ph.D. is in Meteorology from Texas A&M University. He has served in a diverse set of positions in Korea, Nebraska, and Washington DC. Through the years he has supported many experiments and war games.

Valerie Pastor is a Research Programmer at SAIC, Monterey, CA. She has over 19 years of programming experience. She is the principle developer of the GRIB to SEDRIS conversion routine.

A00-37229

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

AIAA-2000-4094

DYNAMIC HYSTERESIS MODELING

P. R. Dahl* and J. H. Ly*

ABSTRACT

This paper presents an improved model of the Dynamic Hysteresis Model (referred to as the DHM) that has previously been developed. We are casting it in Math Works MATLAB language, which is similar to C, and which is functional in Math Works SIMULINK computer program. A background discussion is given of some earlier history of dynamic hysteresis modeling related mostly to studies of friction. The paper then gives a complete rationale of its invoked Prandtl Laws and their application to the DHM and postulates a new Law needed to describe reversing friction and other hysteretic processes. The resulting computer code is spelled out in detail that should make it possible to be easily converted to other computer codes. Typical DHM hysteresis loops from a simulation are shown. A model of Stiction that uses the new DHM is presented. The Stribeck effect is discussed and a different model for it is proposed that uses the DHM Stiction model. Examples are provided of a SIMULINK model of a simple spring-mass system with the new DHM Stiction model with added viscous friction. Simulation results are shown that demonstrates stiction and the Stribeck effect.

* Members of the Technical Staff, The Aerospace Corporation, P.O. Box 92957, Los Angeles, CA 90009-2957
Phil.R.Dahl@aero.org, Jason.H.Ly@aero.org

Copyright © 2000 by P. R. Dahl. Published by The American Institute of Aeronautics and Astronautics Inc., with permission.

1. INTRODUCTION

The need for modeling of hysteretic processes frequently arises in the formulation of dynamic systems that are to be simulated on computers. Applications for the hysteresis models used by the authors include; bearing friction, electric cable harness friction, magnetic drag torque in electric motors, piezo-electric actuator motion vs. voltage, visco-elastic dampers, sliding friction, and magnetic materials B-H characteristics. The so-called Dahl model[1], which was the solid friction model or SFM originated in 1968, formed the basis of a simple hysteresis model for solid friction which was used in many of these applications.

The terminology, solid friction, is used to describe the frictional effects of solid materials that behave dynamically as a function of the sign of velocity and its integral, the displacement. Hence the direction of applied motion and velocity effect the solid friction dynamics but the magnitude of velocity wont directly effect solid friction as it does in fluid friction.

The addition of another state variable, df/dt, provided the means to introduce a stiffness to Coulomb friction at the rest position where the velocity went to zero. The SFM thereby includes solid materials elastic

and plastic properties but not fluid viscous properties. Thus the fluid materials in the friction process are assumed to be without elastic and plastic properties.

The fluid friction model or FFM is a function of the first power of velocity for fully developed linear laminar flow and can be added in parallel to the SFM or the DHM. An example would be the aerodynamic laminar flow friction drag of a landing aircraft acting in parallel with the solid sliding friction brake drag and the tire rolling friction. If the aerodynamic drag is turbulent, the FFM should be related also to the velocity squared. If the lube or liquid or gas did possess some elastic-plastic properties, it was assumed that they would be included in a separate solid friction model in parallel with the primary SFM and the FFM. This is the way that visco-elastic properties can be modeled.

Static friction is a terminology that is used to describe the stickiness of an object to a surface that prevents any motion before a threshold applied force is reached that will cause the start of motion or breakaway from the stuck position. Prior to the SFM, the stuck at rest hypothesis did not accomodate a rest stiffness in any physically believable way.

Kinetic friction is a terminology that is used to describe friction as a function of velocity and is usually characterized by velocities above fully developed Coulomb friction velocities but not zero velocity. It also is used to characterize friction at higher velocities where fluid dynamical effects predominate and solid friction effects may disappear due to fluid interfacial static pressure or dynamic lifting of the friction surfaces.

The authors believed that static and kinetic friction could be encompassed within the same solid friction description framework as the SFM and the DHM. Since static and kinetic friction, when they occur together at low velocities, is referred to as stiction (or sticktion), this phenomenon ought to fall into the solid friction framework. A stiction model was developed by Dahl[2] in 1984 that uses two SFM's with opposing friction forces in parallel. It was tested on data obtained at Lockheed Missile and Space Co. on the Space Telescope Program, however this work was never published in the open literature. The model presented in this paper is used to demonstrate the efficacy of this stiction model.

The combination of solid and fluid friction in these early models went a long way toward the simulation treatment of many bearing friction and visco-elastic problems that arose in the aerospace industry. Dahl and Wilder[3] published the Dynamic Hysteresis Model, or DHM, for hysteresis in piezo-electric actuators in 1985. The earlier DHM model applied to piezo-electric materials properties. The DHM went a step further than the simple solid friction model or SFM by incorporating the Prandtl laws that had been postulated and published by Ludwig Prandtl[4] in 1928 and discovered and used by Osbourne and Rittenhouse[5] in the modeling of Control Moment Gyro gimbal bearing friction. The DHM had originally been formulated by Dahl in BASIC computer language and later by others in FORTRAN. It evidently has not enjoyed widespread usage since then, probably due to its perceived non-linear complexity and the lack of need for high fidelity simulation of solid friction type problems.

A great need for high fidelity friction models developed in a broad area of control systems that are also addressed by engineers working on high accuracy servos used in industry and Universities. Increased fidelity was needed to achieve required accuracy in things like robotic control and optical pointing systems. A number of authors[6,7,8,9] made significant contributions to the subject of friction modeling and they partially accepted Dahl's idea of making solid friction dynamically dependent on velocity direction only and not velocity magnitude. Because the SFM and DHM first publications did not stress velocity dependency and did not provide a model for stiction, there was focus of attention by these investigators on those parts of the models that were velocity dependent . This approach permitted the modeling of the Stribeck effect primarily as a function of velocity with less regard to the SFM or the DHM. The research on advancing state space control methodology continued with models that were available and developing for treating the important problem of stiction.

In this paper, we recast the original DHM computer code into the Math Works Company software program MATLAB[10] and it's satellite program SIMULINK. The MATLAB programming language is very similar to C code and can be converted easily. However, the logic flow diagram of the DHM of this paper is somewhat different than the original 1985 paper version. The shape function remains the same and the three Prandtl laws are implemented in the same way. The First Loading Exceedence or FLE rule or law postulated in the original paper was an add-on law that ranked with the Prandtl laws which were considered antecedent to the FLE. The FLE law was found to be necessary in order to preserve the steady

state asymptotic behavior of the underlying fundamental Coulomb friction law. Due largely to the difference in computer languages, testing and debugging of the MATLAB code was very troublesome and so the FLE law was found instead to be needed as antecedent to the Prandtl laws and applicable to both loading and unloading paths. This finding prompted the renaming of the "First Loading Exceedence" or FLE law to the "Yield Point Transition" or YPT law which is felt to better describe the logic in question.

2. THE DYNAMIC HYSTERESIS MODEL

A description of the DHM can be found in the Dahl-Wilder paper[3]. However, due to the time that has elapsed and slightly different interpretations in light of the new MATLAB code, it seems appropriate to provide another. The Prandtl laws follow.

I. The Initial Slope (IS) Law, Prandtl Law 1*

Immediately after reversal of the sense of deformation, the slope of the stress-strain diagram has the same value as at the beginning of the first loading.

II. The Same Shape (SS) Law, Prandtl Law 2

The shape of any branch of the stress-strain diagram is uniquely determined by the position of the point where the last reversal of the sense of deformation occurred.

III. The Loop Closing (LC) Law, Prandtl Law 3

If the sense of deformation is not reversed again, any such branch will pass through the point where the last but one reversal of the sense of deformation occurred; thereafter, the stress-strain diagram continues as if the loop had never been formed.

IV. The Yield Point Transition (YPT) Law

If all known loops comprised of elastic type paths have been closed, and the current path is still elastic in character, the reversion to the first plastic loading curve will occur when the stress exceeds the current "true" yield stress. The new yield stress will increase along with the increasing load stress and will be remembered as the "true" yield stress.

Example: A fully annealed pure material such as copper that has not been strain or work-hardened will follow a plastic stress-strain curve when loaded in tension. In the loading process, it becomes work-hardened up to the stress level where the load is removed or the direction of loading is reversed. The maximum stress level at the point of load removal becomes the new "true" yield point. A subsequent loading will follow a steeper and apparently more linear elastic path as the specimen is loaded up to the yield point at which point it will revert to the previous plastic path. This behavior is shown in Figures 1 and 2. This is essentially the same statement that was postulated for the FLE law in the '85 paper. The YPT Law in this paper is a restatement and clarification of the FLE Law and the coding of it for MATLAB required a similar but different mechanization that results in a greatly improved code. The YPT law is elaborated upon more here because of its subtlety. Upon examination, it can be regarded as another statement of Prandtl's third law that applies to the initial plastic stress-strain path and to the large reversals in loading deflections that occur in engineering friction problems.

3. MATH MODELING OF THE FOUR LAWS

The order in which the Laws are explained proceeds along a path that perhaps best explains the DHM.

The Same Shape Relation (SS)

This describes the fundamental shape of the full loading (or full unloading) of the friction force vs. deflection curve. It was the starting point of the DHM development that used the SFM function

$$df/dx = sigma*(1 - sgn(xdot)*(f/fc))^i \qquad (1)$$

where:

sigma = the stiffness or slope at f=0
xdot = velocity
fc = asymptotic value of f, the Coulomb level

In this slope function, the initial position, $x(t=0)$, and initial force $f(t=0)$ were assumed zero. The

* The English wording here of the three following laws is that taken from the translation of the Prandtl paper in German in Reference 4 by Osbourne and Rittenhouse in Reference 5.

resulting initial slope was equal to *sigma*. At later velocity reversals, the function *sgn(xdot)* would change sign and the slope would be different from *sigma*, however, the shape of the curves after each turnaround would be the same with the curve approaching it's asymptotic value of *fc* for *xdot>0* and *–fc* for *xdot<0*. This same shape feature is to be preserved in the DHM model. The shapes of Eq(1) are described at length in another article by Dahl[11].

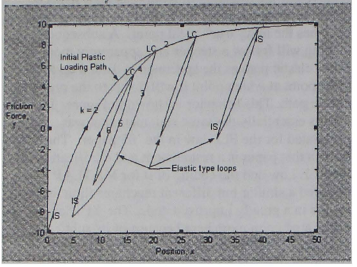


Fig. 1 Characteristics of the DHM and its Laws

Re-Initializing Stiffness (IS)

The stiffness after turnaround is initialized at the turnaround point by storing the value of *f* at turnaround and subtracting that value *f(k)* from *f* in Eq(1). This makes *df/dx* independent of *f* at the turn point. The new DHM then takes the form

$$df/dx = P*(2-sgn(xdot)*(f-f(k))/fc)^e \quad (2)$$

where:

P = sigma
e = i
k = turnaround and path index number
f(k) = value of *f* at kth turnaround.
The Prandtl slope is now *P*2^e*, which is the slope for all turnarounds and also for the initial first loading curve or path starting at *+fc* or *–fc*. For the purposes of describing friction, the asymptotic values *fc* and *–fc* are regarded as the initial starting points for new "initial" loadings for either *xdot>0* or *xdot<0* respectively. "Initial" loadings may be considered turnaround points if the value of f immediately prior to turnaround is very close to the asymptotic value.

The symbol *e* is replacing *i* that was used in the original SFM formulation.

k Path Indexing

The path index *k* denotes a numbered turnaround point where the velocity changes sign and the direction of motion reverses. A computer run is initialized by the sign of velocity where, for example, *xdot* is positive and tensile loading is taking place. In this case, *k* is initialized arbitrarily with the value 2 in the MATLAB code for reasons of avoidance of computational difficulties. If *xdot<0* initially, compressive loading is taking place and the value of *k* is set to 3. For each turnaround, the index is incremented and used as a storage location in memory for the value of *f* at the turnaround point which is set to the value *f(k)*.

Loop Closure (LC)

Because of the IS and SS laws, an upward (tensile) *k* path will cross the immediately preceding *(k-1)* path at the point where *f = f(k-1)*. Because this point can't be hit exactly in a computer simulation, the criterion for ascertaining the loop closure point is to declare loop closure when

$$|f - f(k)| >= |f(k) - f(k-1)| \quad (3)$$

that is, when the length of the current path equals or slightly exceeds that of the preceding path. When this occurs, the Prandtl third law pertains and the loop is forgotten by decrementing the *k* index by 2. The loops that are closed with path *k* values greater than 2 or 3 are referred to as minor loops.

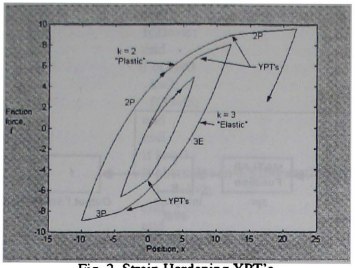


Fig. 2 Strain Hardening YPT's

Yield Point Transition (YPT)

Strain hardening takes place in a material (also called work hardening) when loading is increased beyond its yield point. Prior to reaching the yield point, the stress-strain curve is said to be in its elastic region, and after the yield point the deformation is said to be plastic. The yield point is defined as the point of transition from elastic to plastic deformations. The minor loops described herein correspond to the elastic loading and unloading paths that the friction process is undergoing in back and forth motions and the point where the minor loops are closed may be thought of as yield points.

Prandtl, no doubt, made these observations in arriving at his laws for solid material properties. It is believed that Prandtl based his third law on the idea that it applied to yield points where the minor loops are closed and where, after closure, the path becomes plastic and therefore it applied to all loop closure points including the "true" yield point. However, if the last loop closure point stress value was less than the yield stress, the subsequent path is still elastic until the yield stress is exceeded, at which point the subsequent path becomes truly plastic and reverts to the initial loading curve. For this reason we incorporated the "First Loading Exceedence" (FLE) rule in the DHM in the 1985 paper[3].

In this paper, we regard the rule as an extension of Prandtl's Laws and refer to it as the fourth Law of "Yield Point Transition".

If $|f| >= fy$ *then set* $f(k) = -fc * sgn(xdot)$ (4)

We can now see that the initial loading curve is truly plastic in the DHM model. There can be numerous unclosed minor loop paths in the case where the amplitude of x motion oscillations of loading and unloading is decreasing, as in a damping situation, and with the k index incrementing by one at each turnaround, the value of k can conceivably get quite large. On the other hand, if k had previously gotten large and the path lengths small, a subsequent increase in amplitude of x motion will cause the unclosed minor paths to increase in length. This causes the minor loops to close and the k values to decrement by two at each closure.

As long as k is greater than two in tensile paths, and greater than three in compressive paths, the "true" yield point will not have been reached. That is to say that the material has at some point in the past been work-hardened to the highest "true" yield point, which can be described by the yield stress, or in our case of friction, the highest "true" yield friction force which is defined as fy. If the amplitude of motion gets large enough that the last minor loop is closed, the elastic to plastic transition still may not have occurred at that point since the friction force may still be less than fy.

The yield point is exceeded only when f becomes larger than fy and work hardening again commences along the original plastic deformation curve. New increasing values of fy will then be continually set at each integration step until the motion is again reversed. A switch to the original plastic deformation curve is made at the point when $f > fy$ and then $f(k)$ is set equal to $-fc$ to revert to the plastic curve in tension when $xdot>0$, or $f(k)$ is set equal to $+fc$ to revert to the plastic curve in compression when $xdot<0$. Since the original plastic curves are asymptotic to the Coulomb friction values, the necessary condition of convergence to these values is satisfied.

4. DHM SIMULATION IN SIMULINK

The Simulink diagram for the DHM is shown in Figure 4. The MATLAB .m function code takes velocity as an input, although it only needs the sign of velocity (the code detects it) to compute df/dx, and multiplies df/dx by velocity to compute df/dt which is the output of the function block. The output of the function block, df/dt, is fed to an integrator which then integrates to obtain the output friction force, f. The variable f is then fed back to the Mux block so that the m function is a function of the variables v and f as

Equation (2) prescribes. The model also takes the constant values P, and fc, as initial conditions fed into the Mux.

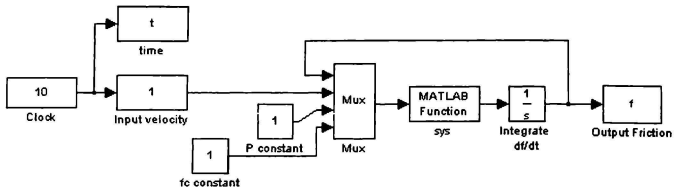


Figure 3. Simulink Diagram of The Dynamic Hysteresis Model

5. THE SIMULINK .m FUNCTION CODE

The code for the SIMULINK.m file function, denoted as 'sys', follows.

```
                                    %sys Defined
function sys = dhm2(u)              %u(1) is friction force.
persistent  k f fc P vold fy flag   %u(2) is velocity or xdot,
                                    %u(3) is stiffness or P,
                                    %u(4) is Coulomb level of friction,  fc.

                                    %Initial Conditions
if flag ~= 1010                     %Flag is zero & set to 101010 end of first step
  P= u(3);                          %sigma set to u(3) which is externally set
  fc = u(4);                        %Coulomb friction level set to external value
  e = 1.0:
  if u(2)>=0                        %  See if velocity is positive
    k=2;                            %if so, set increasing plastic curve value = 2
  end
  if u(2)<0                         %See if velocity is negative
     k=3;                           %if so, set decreasing plastic curve value =3
  end
  f(3)= fc;                         %Initial decreasing plastic start pt is +fc
  f(2) = -fc;                       %Initial increasing plastic start pt is -fc
  err = fc/400;                     %max undershoot error at loop closure
  fy=0;                             %Dead soft material has zero yield point
end

                                    %Start of the main program
if sign(u(2))==sign(vold)           %See if on path i.e. velocity didn't change sign?
     if abs(u(1))>fy                %If not, did f exceed yield pt value fmax
     fy = abs(u(1));                %If so, set new yield point
    if u(2)>=0                      %and if velocity is positive
```

```
    k=2;                                %set k to increasing plastic curve value = 2
    f(2)=-fc;                           %and set unconfirmed turn pt 2 value to -fc
    end
   if u(2)<0                            %If velocity is negative
     k=3;                               %set k to decreasing plastic curve value = 3
     f(3)=fc;                           %and set unconfirmed turn pt 3 value to +fc
    end
    end
    if k>=3                             %If path k value is >3, we're on an elastic path
    if abs(u(1)+err-f(k))>=abs(f(k)-f(k-1))   %so check to see if a loop is closed,if so
        k=k-2;                          %forget loop and revert to k-2 path
    end
    end
end


if sign(u(2))+sig==0                    %See if there is a velocity sign change
    k=k+1;                              %and if so, we're on new path, k+1
    f(k) = u(1);                        %and so store confirmed turn pt k+1 value of f
end
flag=1010;                              %Not an initial condition step from here on
vold= u(2);                             % store sign of velocity this step for next step.

                                %Compute value of friction derivative

df_dt = u(2 )*P*((2 - sign(u(2))*(u(1) - f(k))/fc))^e;
sys = [df_dt];                          %Pass sys function value to integrator

                                %Done, go to first line for next step
```

6. SIMULATION EXAMPLE OF THE DHM

An example of the "static" and dynamic responses of the DHM to an arbitrary prescribed input velocity is described at this point. The SIMULINK diagram used for this example is the one shown in Figure 3. The values of P, and fc are input constants, and e is included in the sys function. The input velocity driver comes from the Fcn block that in turn is driven by the time generating clock. The output has a viscous friction component that can be added to the DHM friction. It can be set to zero with a gain block. The driving input velocity function of time is

$$xdot = (10 - 0.5*t)*sin(2*pi*t/10) \qquad (5)$$

and its integral, the position, is shown in Figure 4, which depicts the dynamic time responses of x and the friction force, f.

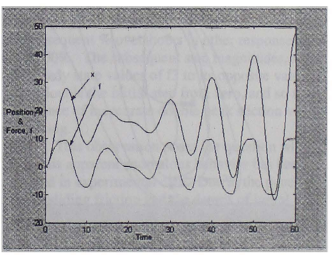


Fig.4 Time Histories of Input Position and Output Friction Force

Figure 5 shows the DHM solid friction hysteresis loops wherein the DHM friction force vs displacement loops are quasi static in that they are independent of the magnitude of velocity but are dependant on the magnitude of displacement. Hence, they are referred to as "static" type functions although this distinction is arguable. The paths in Figure 5 are denoted by their k values.

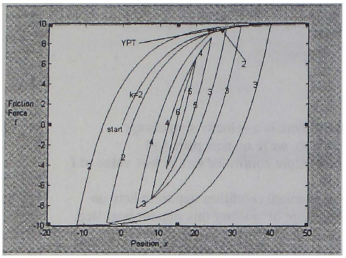


Fig. 5 Simulink DHM Paths. F vs x

As the velocity oscillation amplitude initially decreases, the path k values increase to a value of 6. The input velocity amplitude increases and the k values decrease by 2 each time the loops close.

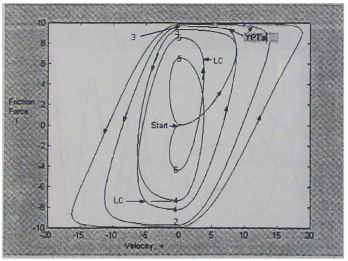


Fig. 6 Simulink DHM Paths, f vs v

Turn points cause the k index to increase by one. The YPT points are not conspicuous but they are annotated where convenient. They have corresponding previous turn points that had pre-turn paths that were plastic and post-turn paths that were "elastic", which defined a new "true" yield point on the opposite loading curve.

Figure 6 shows a similar plot of DHM friction hysteresis loops with friction force plotted vs velocity. The turn points are seen to be along $v = 0$. These loops are analogous in part to the B-H hysteresis loop in magnetic materials[12].

7. A STICTION MODEL

As mentioned in the Introduction, a model for Stiction is introduced in this paper. It is based on an earlier model that was motivated and postulated in 1984 by Dahl and tested for efficacy on data from W. Haile[2]. The concept of differencing two SFM's stems from the observation that the response of a lead circuit to a step input gives an overshoot to its steady state output. This concept was also independently postulated by Bliman and Sorine[8] in 1991 and aptly described by Olsson et.al.[9]. The lead circuit can be mechanized by differencing two lag circuits, which are analogous to two SFM's, or now, two DHM's with the exponent e set equal to 1. The step response for zero initial conditions will be,

$$f(x) = fca*(1-exp(-x/xca)) - fcb*(1-exp(-x/xcb)) \quad (7)$$

Where fca and fcb are the Coulomb friction values and $xca = fca/Pa$ and $xcb = fcb/Pb$ are the characteristic displacements. If we assume $xca << xcb$, Eq.(7) becomes,

$$f(x) = (fca-fcb) + fcb*exp(-x/xcb) \quad (8)$$

Thus the shape of the overshoot response is easy to visualize and predict as long as the assumptions hold. The peak stiction equal to fca occurs near $x = 0$ and the friction decreases exponentially to $(fca-fcb)$ with a distance constant of xcb. It is curious that the steady state friction level could be negative if fcb were to be greater than fca. This obvious error might be corrected by changing the sign of the input to the dual model. However, there would be a possible negative undershoot for this strange combination of parameters. The shape of function $f(x)$ for a plausible set of parameters, however, follow the Stiction test data

observed elsewhere[13]. We shall see shortly this type of response in a SIMULINK simulation.

8. THE STRIBECK EFFECT

The Stribeck friction effect has been modeled as[6],

$$f(v) = fc + (fs - fc) * exp(-|v/vs|^2) + fv*v \quad (9)$$

which has somewhat the graphical appearance of Eq.(7) except it is a function of velocity rather than position and it increases monotonically with velocity by virtue of the last term. If the *fc* term is considered to be a DHM then the other terms are added as a FFM. This makes sense if the Stribeck effect terms reflect steady or slowly varying velocities. If not, and the Stribeck effect includes slow to rapidly varying velocities such as in a frequency sine sweep, then the DHM stiction model may be appropriate since it can capture time response behavior by its dynamic character. The model to simulate the Stribeck effect that is proposed and that will be studied here is given by

$$dfa/dt = v*Pa*(2-sgn(v)*(fa-f(ka))/fca)^ea \quad (10)$$

$$dfb/dt = v*Pb*(2-sgn(v)*(fb-f(kb))/fcb)^eb \quad (11)$$

$$F = fa - fb + B*v \quad (12)$$

The *fa* and *fb* friction variables are those described in Section 8 above and *F* is the combined Stiction and viscous friction force.

9. SPRING-MASS-SLIDING FRICTION EXAMPLE

A dynamic system that embodies the hysteretic characteristics discussed in this paper is simply modeled by the spring-mass-sliding friction problem shown schematically in Fig.(7).

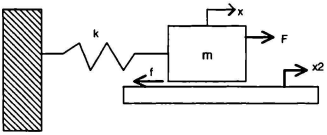


Fig. 7 A System with Hysteretic Friction

The corresponding Simulink schematic diagram is shown in Fig.(8). The mass is assumed to be 1 lb and a spring was chosen to produce a natural frequency of 5 hertz, which yielded reasonable and interesting plots. The two DHM's are shown with different distance constants of 1e-4 ft and 1.2e-3 ft and with fc's of 0.10 lb and 0.06 lb respectively. The stiffer DHM has the usual negative feedback and the less stiff and longer distance constant DHM has positive feedback. These values were chosen to show somewhat reasonable results. A viscous friction component has been added to the DHM's in parallel with adjustable gain. A small value of 0.01 is shown in the diagram that has little effect on results.

Input driver functions can be applied as a force input applied to the mass, *F*, or as an independent velocity input to the base that the mass is sliding on. Results from the simulation are presented in Figs. (9 – 12).

The Stiction Response

Fig.(9) shows the x2 response of *(fa – fb)*, which is f3 in the simulation, to a step input in velocity to the base, v2, with the mass held stationary. This response was obtained to show the details of the simulation in a way so as to validate the DHM Stiction model. The result shows f3 vs x2, where x2 is the relative motion between the mass and the base. The distance constant of 0.012 ft that was predicted is evident in the response.

The first overshoot stiction level is somewhat less than the *fca* value of 0.10 lb that was estimated. This is due to the prediction assumption being relatively crude. The actual first overshoot was 100% and subsequent % overshoots in other responses are the same 100%. The subsequent step magnitudes, from near steady state values of f3 to its opposite value, are nearly double the initial step from zero, and so the appearance of inaccurate double peak stiction values is misleading.

This observation poses the question whether or not this apparent anomalous behavior has been observed in experimental data. Due to the paucity of data on sliding friction and the details of initial loading conditions, the resolution of the question may not be near at hand.

The plot of f3 vs t is not shown because it is identical in appearance to the f3 vs x2 plot due to the input velocity being constant.

The Stribeck Effect Type Response

The Stribeck effect is simulated by connecting a path in parallel with the DHM Stiction model. The path contains a gain block that represents the linear viscous friction coefficient, B. A plot of the response of the total friction, F, given by Equation (12), was obtained and is shown in Fig. 10. It was generated by an input ramp in velocity, v2. The shape of the curve has the characteristics of Eq. (9) except that it has no offset or constant value at zero velocity.

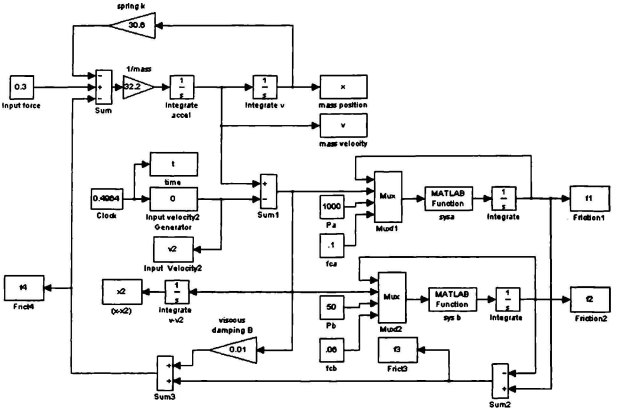


Fig. 8 Simulink Diagram of the Spring-Mass-Sliding Friction Problem

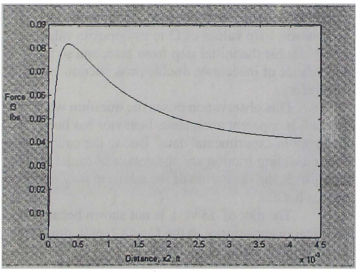


Fig. 9 Stiction Motion Response to Base Input

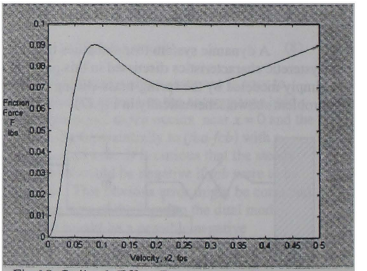


Fig.10 Stribeck Effect Response to a Ramp

The zero value of F for the initial condition when the velocity is zero was borne out by the tests reported to Dahl by Haile[2] as well as the overshoot in the f3 vs x2 type of response. Rabinowicz[13] also shows experimental data that bears out the form of Fig. (9). However, there may be little or no data available that will verify the velocity ramp response behavior of Fig.(10).

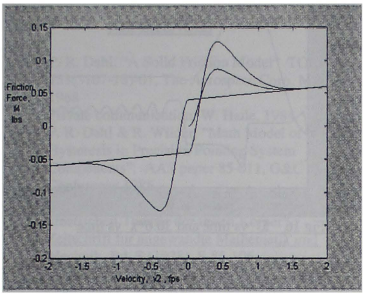


Fig.(11) Transient Stribeck Effect Hysteresis Loops

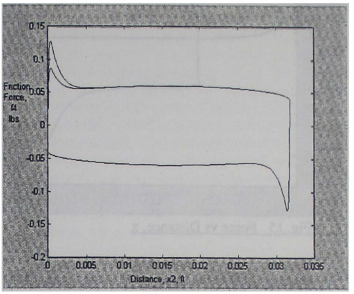


Fig.(12) Stiction Effect Hysteresis Loops

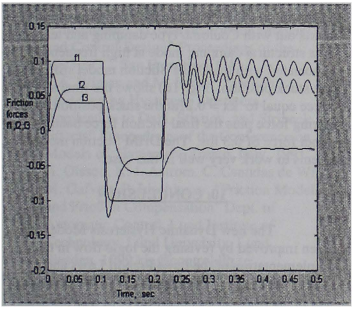


Fig. 13 Response to Step in Force Input

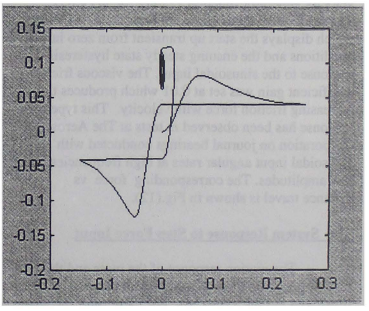


Fig. 14 Force vs Velocity, v

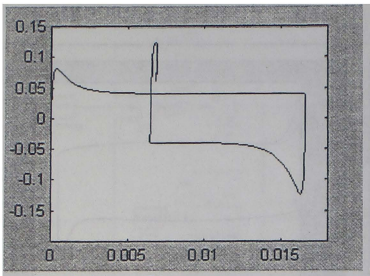


Fig. 15 Force vs Distance, x

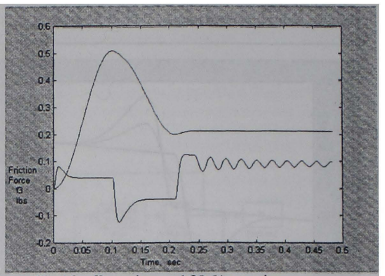


Fig 16 f3 vs time and 30.6*x vs time

An alternative test to the ramp input is the application of a sinusoidal input, which is easier to instrument and implement in a test set up. Also, the steady state cyclic response can be more readily adjusted and results more easily obtained. This test was simulated using the above Simulink diagram with the input velocity generator function *2*sin(2*pi*u(1)/.05).* The result obtained is shown in Fig.(11) as f4 vs v2 which displays the start up transient from zero initial conditions and the ensuing steady state hysteresis response to the sinusoidal input. The viscous friction coefficient gain was set at 0.01 which produces the increasing friction force with velocity. This type of response has been observed in tests at The Aerospace Corporation on journal bearings conducted with sinusoidal input angular rates at high frequencies and low amplitudes. The corresponding force vs distance travel is shown in Fig.(12).

The System Response to Step Force Input

The motion response of the mass and the resulting friction force to a step input of force is illustrated in Figs. (13)-(16). For these runs, the base was locked in place by setting v2 = 0. The input force applied was a step of magnitude 0.3 lb. Fig.(13) shows the friction responses of the two DHM's, f1 and f2 and their difference, f3. The way the f3 overshoot is generated is evident. The final values of f1 and f2 account for the sticking characteristic whereby f2 sticks at –0.04 and f1 sticks at +0.05 and f3 sticks at +0.09 lbs. Figs.(14,15) display the friction force vs velocity and distance hysteresis loops that have the same behavior as indicated in Figs. (11,12) with a damping effect that starts out with Coulomb type damping and terminates in a structural damping mode at high frequency associated with the DHM Stiction model stiffness at the stuck position. Fig.(16) shows the final spring force equal to kx = 0.21at the stuck position. This spring force plus the final stiction force balances the input force of 0.3 lbs. The DHM Stiction model seems to work very well in this case.

10. CONCLUSION

The new Dynamic Hysteresis Model has been improved by revising the logic flow in the implementation of the three Prandtl Laws and a fourth YPT Law. Implementation has been carried out in the MatLab computer .m code and the source code is provided. There can be no guarantee that it will work to the entire satisfaction of the user but it is hoped that it will be a useful tool for the solution of

many problems involving hysteresis. The DHM has been demonstrated to perform extremely well in the examples given in the paper. The DHM Stiction model presented seems to work very well and it is believed that it should enjoy widespread usage. The added viscous friction term to the DHM Stiction model appears to provide a simulation means for some Stribeck Effect type problems.

REFERENCES

[1] P. R. Dahl, "A Solid Friction Model" TOR-158(3107-18)-01, The Aerospace Corp. May 1968

[2] Private communication, W. Haile, 1984

[3] P. R. Dahl & R. Wilder, "Math Model of Hysteresis in Precision Pointing System Components", AAS paper 85-011, G&C Conference 1985

[4] L. Prandtl, "Ein Gedankenmodell zur kinetischen Thoerie der festen Korper", Zeitschrift fur angewandte Mathematik und Mechnik, Bd. 8 (1928), S-85-106

[5] H. A. Osbourne & D. L. Rittenhouse, "The Modeling of Friction and Its Effect on Fine Pointing Control" AIAA Paper No. 74-875, 1974

[6] B. Armstrong-Helouvry Control of Machines with Friction , Kluwer Academic Publishers, Norwell, MA 02061, 1991

[7] C. Canudas de Wit, H. Olsson, K. J. Astrom, P. Lischinsky, "Dynamic Friction Models and Control Design" Proceedings of the 1993 ACC, San Francisco, CA

[8] P.A. Bliman and M. Sorine "Friction Modeling by Hysteresis Operators, Application to Dahl, Stiction, and Stribeck Effects" Prcoeedings of the Conference on Models of Hysteresis, Trento, Italy, 1991

[9] H. Olsson, K. J. Astrom, C. Canudas de Wit, M. Gafvert, P. Lischinsky, "Friction Models and Friction Compensation" Dept. of Automatic Control, Lund Institute of Technology, Lund University, Lund Sweden, 1999 kja@control.lth.se

[10] The MathWorks, Inc. Natick MA 01760 info@mathworks.com

[11] P. R. Dahl, 'Solid Friction Damping of Mechanical Vibrations" AIAA Journal, 14(1):1675-82, 1976

[12] P. R. Dahl, "A Magnetic Hysteresis Model" Guidance and Control 1992, Advances in Aeronautical Sciences Vol. 78, AAS paper 92-016

[13] E. Rabinowicz, "The Nature of Static and Kinetic Coefficients of Friction" Journal of Applied Physics, 22(11):1373-79, 1951

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37230

AIAA-2000-4095

TOWARDS REALISTIC FOREST MACHINE SIMULATORS*

Prof. Dr.-Ing. E. Freund, Dipl.-Ing. M. Krämer, Dr.-Ing. J. Rossmann

Institute of Robotics Research (IRF), University of Dortmund,
Otto-Hahn-Str. 8, 44227 Dortmund, Germany

ABSTRACT

To develop realistic forest machine simulators is a demanding task. A useful simulator has to provide a close-to-reality simulation of the forest environment as well as the simulation of the physics of the vehicle. Customers demand a highly realistic three dimensional forestry landscape and the realistic simulation of the complex motion of the vehicle even in rough terrain in order to be able to use the simulator for operator training under close-to-reality conditions. The key development issues for a forest machine simulator comprise the simulation of the crane handling, the physics of tree cutting and loading process and — the main issue of this paper — the simulation of the vehicle motion on a three dimensional surface in natural sceneries. The realistic simulation of the vehicle, especially in combination with a motion platform under the driver's seat, greatly improves the effect of immersion into the virtual reality of a simulated forest and the achievable level of training of the driver.

INTRODUCTION

The forest machine simulator is based on COSIMIR/VR [1], the 3D robot simulation and virtual reality system of the Institute of Robotics Research (IRF). The simulator provides a 3D view of the virtual forest to the driver. It is being sold as a commercial product e.g. by the harvester manufacturer PONSSE. The simulator supports all training aspects mentioned above and described in adjoint papers [6, 7], but one of the key features that distinguishes the simulator from the competition [2, 3, 4, 5] is its capability to simulate the motion of the vehicle correctly — even in rough terrain. Usually, the major goal of training on forest machine simulators is to improve the driver's capabilites concerning the crane handling and the use of the onboard computer. The training of vehicle guidance skills however is often limited since the simulation of driving features of the vehicle is often only implemented rudimentarily or is limited to plane motion. Nevertheless, the driver's ability to safely guide the vehicle through rough terrain and along steep slopes is absolutely essential for the safe handling of e.g. a harvester or forwarder.

THE FOREST MACHINE SIMULATOR

The kinematics of physical forest machines are adapted to the operation environment and purpose. Therefore they differ from other vehicles concerning controllability and performance. In general the ability

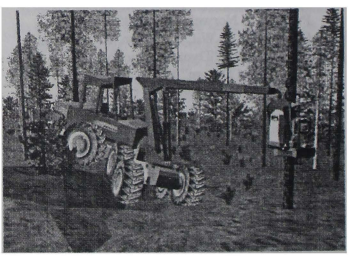


figure 1: Harvester in virtual forestry environment

to drive a vehicle e.g. in traffic cannot be transferred to forest machines because the main goals are different. Whereas the car-driver usually goes for speed and safety under almost unchanged conditions, a driver of a forwarder usually has to deal with changing load conditions and related problems when driving a fully loaded vehicle on steep slopes. The forest machine simulator presented in this paper now for the first time makes it possible to practice safely in

* Copyright © 2000 by IRF, Institute for Robotics Research. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

difficult scenarios, like e.g. harvesting operations on a steep slope, or maneuvering a forwarder in rough terrain. In the presented simulator, the terrain model is not just used for the visual presentation but also to generate data for a motion platform mounted underneath the driver's chair which provides an additional feeling of reality to the driver. This considerably improves the effect of immersion into the "virtual forest". Experience showed that such training of drivers on the simulator with the motion platform is very intensive and close to reality and thus the training results can immediately be transferred to the real forest machine.

Basics of Forest Machine Simulation

The simulation method of *kinematics projection* for vehicle guidance across a curved surface is a general method and has been developed at the IRF. It can be applied to a wide range of different kinematics and will be explained for a model of a forest machine with six wheels. The method combines a high degree of realism and a realtime execution. This fast method allows for a high quality vehicle guidance simulation on a PC platform.

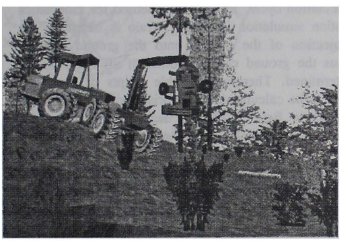


figure 2: Realistic Harvester Simulation in rough terrain

The terrain model for vehicle guidance is based on a digital map in form of a polygon mesh. It describes the topography of the landscape. This information is transformed into a quadtree structure for high speed data access in large terrain models.

The method of *kinematics projection* for vehicle guidance across curved surfaces is a four sequence process (figure 8). Within each simulation cycle the four steps are carried out in succession.

The first step is the simulation of the steering process. Then the locomotion of the vehicle is simulated according to its current dynamic state. Afterwards the 'touch the ground' algorithm projects the kinematics onto the polygon net at the new position and a correction is calculated to make sure that the vehicle 'stays' on the curved surface. Finally, the algorithm recalculates the joint values of all swing axles of the vehicle in order to prevent the wheels from dipping in or lifting off the ground.

ENVIRONMENT MODELLING

The more realistic the virtual forest environment is the more effective is a training session on the simulator. In order to achieve a highly realistic forestry scenery, latest VR technologies are used. For the description of the three dimensional virtual world a description language is used, which has been developed at the IRF for universal virtual reality modelling: EGMOL, the **E**nhanced **G**eometric **M**odeling **L**anguage [1]. Among other specifications it describes the geometry and textural appearance of the virtual objects and their degrees of freedom. Furthermore, display techniques like texture mapping and texture shading are used in order to realize a realistic appearance of the virtual objects.

Moreover, the objects in the virtual forest must be *interactive* in order to be performing a realistic training when grasping or cutting trees or branches.

One of the interesting aspects in this area is the interaction of objects with non-flat ground. In reality e.g. the operator cuts a tree to length and lays it onto the ground. In order to simulate this operation, gravity has to be simulated to make the tree fall realistically and a solid-ground simulation is necessary to simulate the interaction between tree and ground. Equally interesting as a falling tree hitting the ground is the motion of a forest machine in rough terrain.

A forest machine simulation demands models of ground and vehicle beyond textures and shading in combination with corresponding algorithms for object interaction. The following explanations focus on the simulation of the motion of the vehicle describing related algorithms and elements in the environment model.

GROUND MODEL

The terrain model consists of a coherent mesh of convex polygons (e.g. figure 3).



figure 3:Basic ground model

The mesh shown above represents a landscape model of about 100 by 100 meters. A mesh of this kind consist of approximately 2000 polygons. A fast access to polygon data is required for real-time simulation of a forest machine moving on the ground. Therefore the ground information is transformed into a quadtree structure (e.g. figure 4).

The main purpose of the quadtree is to perform high speed access to altitude information for a given

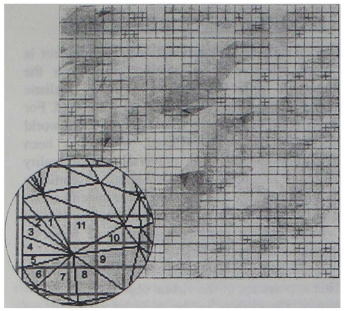


figure 4: Quadtree structured ground model

coordinate (x, y). A list of polygons is assigned to each leaf of the quadtree. The subdivision of the tree is terminated if the number of polygons is less than a given limit. A hit algorithm then determines the corresponding leaf and polygon and calculates the altitude $p'.z$ of the point $P = (p.x, p.y, 0)$.

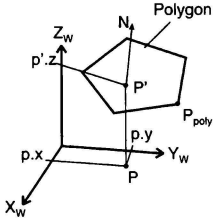


figure 5: Ground altitude derivation

$N = (n.x, n.y, n.z)$ with $n.z \neq 0$ is perpendicular to the polygon and the evaluation of $(P'-P_{poly})N = 0$ derives the ground altitude for P:

$$p'.z = \frac{[n.x(p_{poly}.x-p.x)+n.x(p_{poly}.y-p.y)]}{n.z} + p_{puly}.z$$

If $n.z = 0$ the hit algorithm does not consider the corresponding polygon.

The data access time is determined by the level of the tree and the number of polygons in a leaf. The minimum number of polygons in a leaf is given by the number of polygons which share at least one common point. Further subdivision of the tree makes no sense because there is no possibility to reduce the number of polygons per leaf. In the ground model shown above up to 11 polygons (figure 4) share a common point.

VEHICLE MODEL

For a suitable motion simulation of the vehicle two basic model elements are essentially required. First of all, it is necessary to describe the position and orientation of the vehicle. These six degrees of freedom are represented by the homogenous transformation T_O (figure 7). Secondly the internal degrees of freedom, represented by four joints (figure 6), need to be specified. Therefore a set of revolute joints, in form of Denavit-Hartenberg parameters, describes these internal degrees of freedom. These joints are linked to a kinematic chain. Thus the simulation of complex kinematics, as the presented forest machine, is made easier. Including joints for the rolling of the wheels there are totally 16 degrees of freedom to be handled.

For a realistic simulation of the motion of the vehicle in rough terrain, all these degrees of freedom have to be determined simultaneously. The computing process however only permits a sequential calculation of the joint values. Because of the mutual influence between certain degrees of freedom some of the joint values have to be recalculated after the first calculation cycle. The general method for kinematics motion simulation in rough terrain is based on the projection of the wheels onto the ground surface. Thus the ground contact points of all tires can be determined. Therefore suitable points of each tire have to be calculated. Appropriate points e.g. for the wheels in figure 7 can be derived by varying the corresponding angles ϕ_{rr} and ϕ_{fr}:

$$T_{tire,rr} = T_O T_r T_{\phi r} T_{rr} T_{\phi rr} T_{Rr}$$
$$T_{tire,fr} = T_O T_\varphi T_{\phi fr} T_{fr} T_{\phi rr} T_{Rf}$$

with

$$T_r = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -l_r \\ 1 & 0 & 0 & -h_r \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{\phi r} = (\phi_r \; 0 \; 0 \; 0)_{DH-Params}$$

$$T_{rr} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -w_r \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{\phi rr} = (\phi_{rr} \; 0 \; 0 \; 0)_{DH-Params}$$

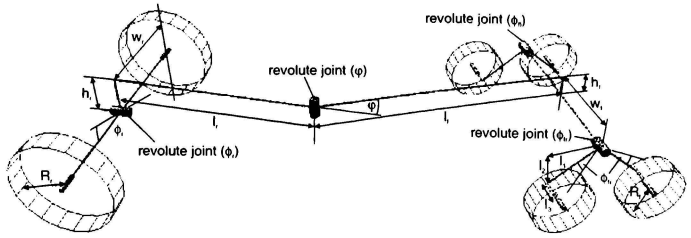


figure 6: Dimensions of the vehicle model to be considered

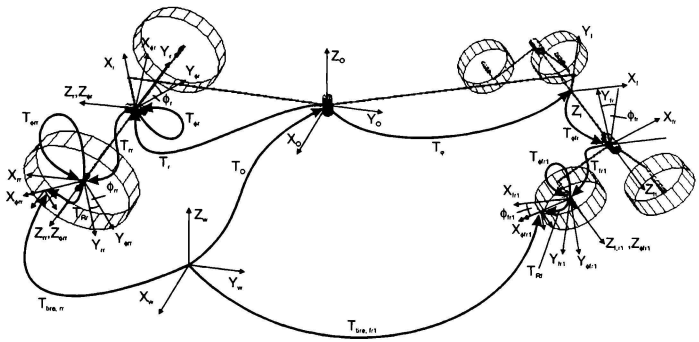


figure 7: Concatenation of homogeneous transformations for potential ground contact point determination

$$T_{Rr} = \begin{bmatrix} 1 & 0 & 0 & R_r \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{\varphi} = \left(\frac{\pi}{2} + \varphi \quad -h_f \quad l_f \quad \frac{\pi}{2} \right)_{DH\,-\,Params}$$

$$T_{\phi fr} = \left(\phi_{fr} \quad w_f \quad 0 \quad 0 \right)_{DH\,-\,Params}$$

$$T_{fr1} = \begin{bmatrix} -1 & 0 & 0 & -l_1 \\ 0 & -1 & 0 & -l_2 \\ 0 & 0 & 1 & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{\phi fr1} = \left(\phi_{fr1} \quad 0 \quad 0 \quad 0 \right)_{DH\,-\,Params}$$

$$T_{Rf} = \begin{bmatrix} 1 & 0 & 0 & R_r \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

THE SIMULATION METHOD OF KINEMATICS PROJECTION

The simulation of vehicle motion in rough terrain is a complex and oscillation sensitive process, particularly because of the interdependence between position and orientation of the vehicle and the internal swing axle joints ϕ_r, ϕ_{fr}, and ϕ_{fl} (figure 6). The center pivot steering of the vehicle makes the motion even more difficult for a realistic simulation. Thus it was absolutely necessary to introduce certain restrictions

and simplifications in order to reduce the complexity of the process:

1. Whereas in reality the single operations of the vehicle (e.g. steering, locomotion, keeping in touch with the ground) occur simultaneously, in simulation they have to be carried out incrementally in succession.
2. The steering and locomotion operation only occur in the x,y plane of the object coordinate system of the vehicle (X_O, Y_O, Z_O in figure 6). Afterwards, position, orientation and the swing axle joint values are corrected to make sure, the vehicle contacts the ground in a realistic way.
3. Because of the already mentioned interdependence of some degrees of freedom, actually the 'touch the ground' algorithm (adapts the position and orientation of the vehicle to the ground and prevents the wheels from dipping in or lifting off the surface) should be executed in a loop. In order to accelerate the calculation speed it is recommended to prevent this loop. Therefore the position and orientation changes per simulation cycle must be tiny so that the algorithm converges sufficiently within the first step of the loop.
4. Low pass filters have to be installed in order to suppress the tendency of position and orientation to oscillate. Equal treatment is required for the internal swing axle joints of the vehicle.
5. In order to prevent the algorithm from sliding the vehicle down a hill and from rotating it towards downhill, the x,y coordinates and the roll angle of the orientation must not be affected by the 'touch the ground' algorithm. The variation of only the z coordinate of the position and the pitch and yaw angles of the orientation adapts the vehicle to the curved surface.

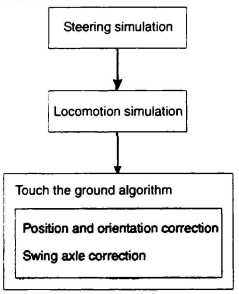


figure 8: The *kinematics projection* simulation method for vehicle guidance in rough terrain

Only these simplifications and restrictions mentioned above reduce the effort for a realistic simulation of a vehicle with 16 degrees of freedom in rough terrain to an acceptable level.

Kinematics projection, the basic simulation method for vehicle guidance across curved surfaces is a four sequence process (figure 8) and is carried out within each simulation cycle.

STEERING SIMULATION

In contrast to simple kinematics as e.g. cars, the position and orientation of a center pivot steered vehicle are affected by the steering process. When changing the steering angle φ by $\Delta\varphi$, the front and rear center of revolution move and rotate simultaneously to their new positions (figure 9).

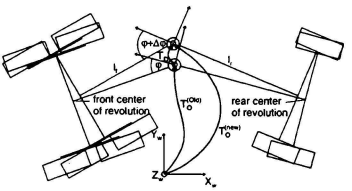


figure 9: Real steering process

According to restriction no. 1 the simultaneous motion of the front and rear center of revolution is divided into a sequence of a translation (Δd_f, Δd_r) and a rotation ($\Delta\theta_f$,$\Delta\theta_r$):

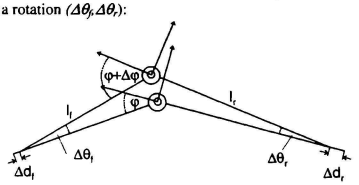


figure 10: Reduction of the steering process to a sequence of translation and rotation

The problem to be solved at this point is the calculation of Δd_f and Δd_r. The resulting state of the vehicle will defer as least as possible from the previous state (principle of minimum energy level). The optimum state is between the two boundary values $\Delta d_f = 0$ and $\Delta d_r = 0$ (figure 11).

An appropriate criterion for state estimation in this context is the translation of the gravity center P_{COG}. As shown in figure 12, the current center of gravity ($P_{COG,\ \Delta df\ =\ 0}$, $P_{COG,\ \Delta dr\ =\ 0}$) can be calculated based on the front axle mass (m_f, $P_{COG,f}$) and rear axle mass (m_r, $P_{COG,r}$):

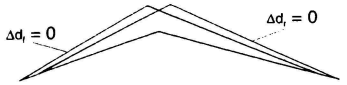


figure 11: Boundary states of the vehicle as a result of no translation of the front or rear center of revolution

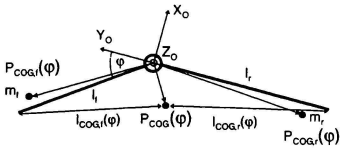


figure 12: Determination of the center of gravity

$$P_{COG}(\varphi) = \frac{m_f P_{COG,f}(\varphi) + m_r P_{COG,r}(\varphi)}{m_f + m_r}$$

Assuming that $\Delta\varphi$ is a tiny angle increment, the center of gravity of each state between the above mentioned two extremes is located approximately on the connecting straight line (gravity line) between $P_{COG,\ \Delta d_f = 0}$ and $P_{COG,\ \Delta d_r = 0}$ (figure 13).

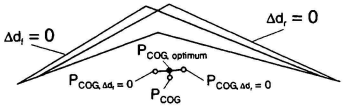


figure 13: Optimum center of gravity

By drawing the perpendicular from P_{COG} to the gravity line yields the gravity center of the optimum state ($P_{COG,\ optimum}$). By means of the position of the gravity center relative to the front and rear axle center of revolution (figure 12 and 14), the transformation

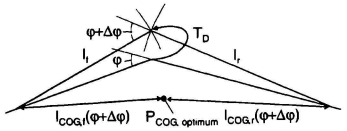


figure 14: New position as a result of the steering process

T_D can be calculated in object coordinates. Then, the new position of the vehicle can be calculated as follows:

$$T_O^{(new)} = T_O^{(old)} T_D$$

LOCOMOTION SIMULATION

After the steering process, the locomotion of the vehicle is simulated. In general ($\varphi \neq 0$) the locomotion process can be approximated by a rotation around M, a momentary center of revolution (figure 15):

$$M = \begin{pmatrix} \dfrac{-l_f}{\sin(\varphi)} - l_r \dfrac{\sin(\pi/2-\varphi)}{\sin(\varphi)} \\ -l_r \\ 0 \end{pmatrix}$$

Considering the velocity v of the vehicle and the interpolation cycle time Δt, the rotation angle $\Delta\alpha$ can be calculated as follows:

rear wheel drive:
$$\Delta\alpha = \frac{v\Delta t}{l_r \dfrac{\sin(\pi/2-\varphi)}{\sin(\varphi)} + \dfrac{l_f}{\sin(\varphi)}}$$

front wheel drive:
$$\Delta\alpha = \frac{v\Delta t}{l_f \dfrac{\sin(\pi/2-\varphi)}{\sin(\varphi)} + \dfrac{l_r}{\sin(\varphi)}}$$

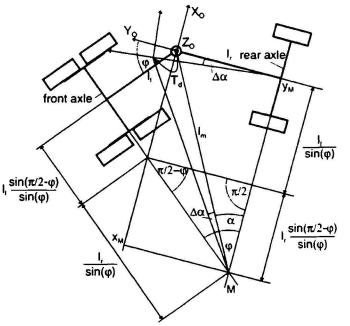


figure 15: Locomotion as rotation around a momentary center of revolution

Then the transformation T_d to the new position of the vehicle can be calculated in object coordinates as follows:

$$R = \sqrt{x_M^2 + y_M^2}$$

$$\alpha = a\tan 2(-y_M, -x_M)$$

$$T_d = \begin{bmatrix} \cos(\Delta\alpha) & -\sin(\Delta\alpha) & 0 & x_M + R\cos(\alpha+\Delta\alpha) \\ \sin(\Delta\alpha) & \cos(\Delta\alpha) & 0 & y_M + R\sin(\alpha+\Delta\alpha) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

If $\varphi = 0$, the locomotion is a simple translation by $v\Delta t$ in Y_O-direction.

TOUCH THE GROUND ALGORITHM

In general, after the simulation of the steering and locomotion process the vehicle has left the ground or the wheels dip into the surface. Thus, a position and joint correction algorithm has to set the vehicle onto the ground surface. The flowchart in figure 17 gives a rough overview of the *touch the ground* algorithm.

Calculation of ground contact points:
First, for each tire a ground contact point has to be derived. A set of n points P_i, $i \in \{1, ..., n\}$, which are located on the edge of a tire, are projected onto the surface (the z-component is set to the ground altitude at the xy-components of the correspondent point):

$$P_{i,G} = \begin{pmatrix} p_i.x \\ p_i.y \\ GroundAltitude(p_i.x, p_i.y) \end{pmatrix}$$

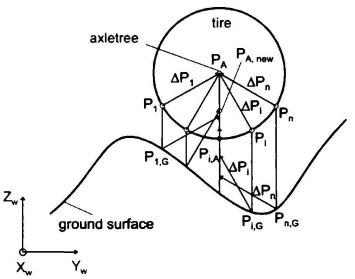


figure 16: Z-axis projection of a tire onto the ground surface

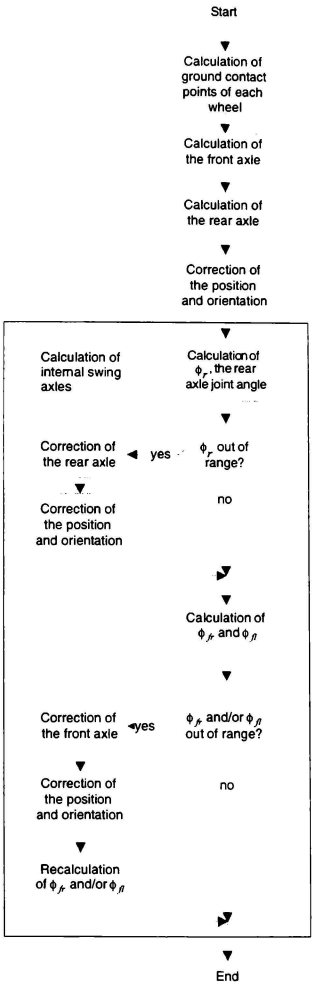


figure 17: Touch the ground algorithm

For each potential ground contact point $P_{t,i}$, an axletree point $P_{t,A}$ is calculated (figure 16):

$$P_{t,A} = P_{t,i} + \Delta P_t$$

All these potential axletree points $P_{t,A}$ only differ in the z-component. The axletree point with the highest z-component results the new approximated ground contact point and represents the new axletree point $P_{A,new}$.

Calculation of the front axle:
Next step to take is the calculation of the front axle of the vehicle. The position and orientation of the front axle can be described by the transformations T_{fr} and T_{fl} to its edge points (figure 7 and figure 18). $P_{A,fr1}$ and $P_{A,fr1}$ are the new axletree points of the two front

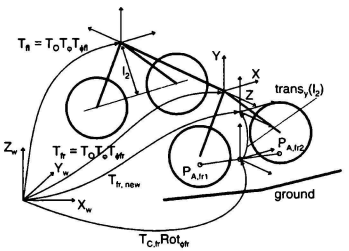


figure 18: Front axle derivation

wheels at the right side of the vehicle, which results from the previous algorithm step. $P_{C,fr}$ is the center between these points.

$$P_{C,fr} = \frac{P_{A,fr1} + P_{A,fr2}}{2}, \quad T_{C,fr} = \begin{bmatrix} 1 & 0 & 0 & p_{C,fr}.x \\ 0 & 1 & 0 & p_{C,fr}.y \\ 0 & 0 & 1 & p_{C,fr}.z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The right edge point of the front axle results from the new center point $P_{C,fr}$ and the current orientation of the right swing axle coordinate system, described by $Rot_{\varphi fr}$:

$Rot_{\varphi fr} = Rot(T_O T_\varphi T_{\varphi fr})$

with

$$Rot(T) = \begin{bmatrix} t_{11} & t_{12} & t_{13} & 0 \\ t_{21} & t_{22} & t_{23} & 0 \\ t_{31} & t_{32} & t_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } T = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The new right edge point $T_{fr,new}$ of the front axle can then be calculated as follows:

$$T_{fr,new} = T_{C,fr} Rot_{\varphi fr} trans_y(l_2)$$

with

$$trans_y(l_2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The left edge point $T_{fr,new}$ of the front axle is calculated analogous.

Calculation of the rear axle:
The position and orientation of the rear axle is determined by the transformations $T_{A,rr}$ and $T_{A,rl}$ to its edge points. $P_{A,rr}$ and $P_{A,rl}$ are the new axletree points of the two rear wheels resulting from the latest ground contact point calculation.

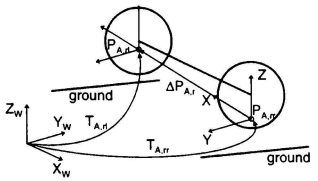


figure 19: Rear axle derivation

$T_{A,rr}$ can be calculated as follows:

$$\Delta P_{A,r} = P_{A,rl} - P_{A,rr} = \begin{pmatrix} \Delta p_{A,r}.x \\ \Delta p_{A,r}.y \\ \Delta p_{A,r}.z \end{pmatrix}$$

$$\alpha = a\tan 2(\Delta p_{A,r}.y, \Delta p_{A,r}.x)$$

$$\beta = a\tan 2(\Delta p_{A,r}.z, \sqrt{\Delta p_{A,r}.x^2 + \Delta p_{A,r}.y^2})$$

$$T_{A,rr} = \begin{bmatrix} \cos(\alpha)\cos(\beta) & -\sin(\alpha)\cos(\alpha)\sin(\beta) & p_{A,rr}.x \\ \sin(\alpha)\cos(\beta) & \cos(\alpha) & \sin(\alpha)\sin(\beta) & p_{A,rr}.y \\ -\sin(\beta) & 0 & \cos(\beta) & p_{A,rr}.z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The left edge point $T_{A,rl}$ of the rear axle is calculated analogous.

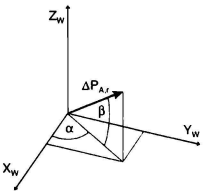


figure 20: Rear axle orientation

Correction of the position and orientation of the vehicle:

After the calculation of the edge points of the front and rear axle the position and orientation correction can follow. The new 'ground contact' position $T_{O,new}$ of the vehicle is calculated as follows:

$$T_{O,new} = T_{Ah,fm} Rot^{-1}(T_O) R_{new} T_{lf} rot_y(\varphi) rot_\Delta$$

with:

$$T_{Ah,fm} = T_{A,fm} Rot(T_O) trans_y(hf)$$

$$T_{lf} = trans_x(c_{stretch} l_f)$$

$$T_{A,fm} = \begin{bmatrix} 1 & 0 & 0 & p_{A,fm}.x \\ 0 & 1 & 0 & p_{A,fm}.y \\ 0 & 0 & 1 & p_{A,fm}.z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_{A,fm} = \frac{P_{A,fr} + P_{A,fl}}{2}$$

$$R_{new} = \begin{bmatrix} X_{new}.x & Y_{new}.x & Z_{new}.x & 0 \\ X_{new}.y & Y_{new}.y & Z_{new}.y & 0 \\ X_{new}.z & Y_{new}.z & Z_{new}.z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$X_{new} = \frac{Y_{new} \times Z_{new}}{|Y_{new} \times Z_{new}|}$$

$$Y_{new} = \frac{D_f}{|D_f|}$$

$$Z_{new} = \frac{D_{rf} \times D_f}{|D_{rf} \times D_f|}$$

$$D_f = P_{A,fl} - P_{A,fr}$$

$$D_{rf} = P_{Ah,fm} - P_{Ah,rm}$$

$$c_{stretch} = \frac{|D_{rf}|}{\sqrt{l_f^2 + l_r^2 - 2 l_f l_r \cos(\pi - \varphi)}}$$

$$rot_\Delta = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation $T_{Ah,fm}$ results from the center point $P_{A,fm}$ between $P_{A,fr}$ and $P_{A,fl}$ of the front axle, the current orientation of the vehicle Rot(T_O) and the translation in y direction by h_f. An analogous procedure supplies $P_{Ah,rm}$. $P_{Ah,fm}$ and $P_{Ah,rm}$ are located within the XY-plane of the vehicle. Therefore by means of D_f and D_{rf} the new orientation of the front axle R_{new} can be calculated. Starting with $T_{Ah,fm}Rot^{-1}(T_O)R_{new}$ in combination with a translation by $c_{stretch}l_f$ in negative X-direction followed by two further rotations supplies the 'ground contact' position of the vehicle. But according to restriction no. 5 only the z component and the pitch and yaw angles may be affected by this procedure, particularly

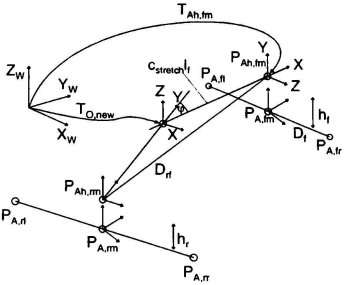


figure 21: Recalculation of position and orientation

since the front and rear axle of the vehicle are stretched by the projection onto the ground surface.

Calculation of ϕ_r, the rear axle joint angle:

After position and orientation of the vehicle are corrected, one wheel of the rear axle dips into the ground and the second wheel lifts off the ground: The distance of each wheel to the ground is approximately

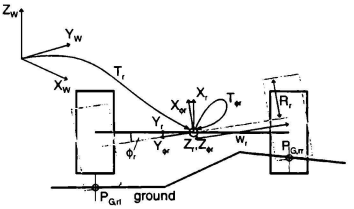


figure 22: Correction of the rear swing axle

the same. By means of the ground contact points $P_{G,rl}$ and $P_{G,rr}$ the appropriate swing axle joint ϕ_r can be determined by the evaluation of the z—component of the following system of equations:

$$P_{G,rl} = T_r T_{\phi r} trans_y(w_r) trans_x(R_r) \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

With

$$T_r = \begin{bmatrix} t_{r,11} & t_{r,12} & t_{r,13} & t_{r,14} \\ t_{r,21} & t_{r,22} & t_{r,23} & t_{r,24} \\ t_{r,31} & t_{r,23} & t_{r,33} & t_{r,34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$k_1 = R_r t_{r,23} + w_r t_{r,32}$$
$$k_2 = R_r t_{r,32} - w_r t_{r,23}$$
$$k_3 = p_{G,rl,z} - t_{r,34}$$

the problem is reduced to a quadratic equation :

$$0 = (k_2^2 + k_1^2)\cos^2(\phi_r) - 2k_1 k_3 \cos(\phi_r) + (k_3^2 - k_2^2)$$

Then the evaluation of this quadratic equation yields the joint value for the rear axle. If this value is out of range, the joint is set to the limit of the range and afterwards a rear axle correction is carried out followed by the recalculation of the position and orientation of the vehicle.

The calculation of the front swing axle joints ϕ_{fr} and ϕ_{fl} can be derived analogous applying the same method.

THE IRF FOREST MACHINE SIMULATOR IMPLEMENTATION

At the IRF, the forest machine simulator has been implemented and the algorithms presented before, prove to provide a highly realistic impression for the driver of the simulated vehicle. This sensation of reality has been enhanced by stereoprojection technology which provides the driver with a 3D view of the scene.

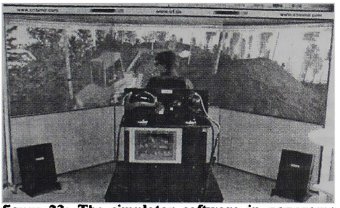


figure 23: The simulator software in panorama view

Fig. 23 shows a driver in his seat running the harvester in the 3D panorama projection environment. In this virtual reality presentation facility, developed at the IRF, the driver is presented a 3D view into the forest on 3 coordinated screens to cover his full field of view. This realization is built using the COSIMIR virtual reality system (www.cosimir.com) which supports the distribution of the computational load on multiple PCs. The new COSIDISC architecture automatically coordinates — almost — arbitrary many PCs to cooperatively display virtual worlds on multiple screens. The panorama projection shown in fig. 23 and fig. 24 is driven by 6 PCs. Each of the 3 screens is driven by 2 PCs, one generating the right eye view and one generating the left eye view for the 3D presentation of the virtual world.

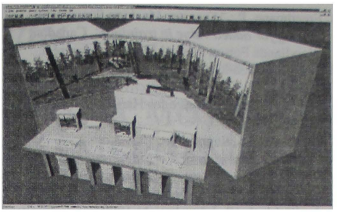


figure 24: The configuration of the 3-screen panorama projection

CONCLUSION

Virtual reality technology today has reached an impressive standard and now is ready to be applied to sophisticated simulation problems. Especially as the technology is now available on standard PCs, providing 3D views into virtual worlds on multiple screens using consumer projection equipment and thus cutting down the prices for realizations by a factor of 10 to 100. This paper describes some of the key technologies required to realize a simulator for forest machines to provide the new drivers with a realistic training facility. The realized simulator provides a close-to-reality simulation of the forest environment as well as the simulation of the physics of the vehicle. Further emphasis was laid on a highly realistic three dimensional forestry landscape and the realistic simulation of the complex motion of the vehicle even in rough terrain in order to be able to use the simulator for operator training under close-to-reality conditions. The key development issues for a forest machine simulator comprised the simulation of the crane handling, the physics of tree cutting and loading process and — the main issue of this paper — the simulation of the vehicle motion on a three dimensional surface in natural sceneries.

Today, this simulator is available as commercial product, to help drivers to learn how to work safely, efficiently and ecologically sensible.

REFERENCES

[1] Freund, E.; Roßmann, J.; Uthoff, J.; van der Valk, U. : "*Towards realistic Simulation of Robotic Workcells*", Proceedings of the IEEE/RSJ/GI Intelligent Robots and Systems, IROS, München, Deutschland, September 1994.

[2] Freedmann, P.; Papadopoulos, E.; Lapointe, J.-F. : "*A Computer-Based Training Environment for Forestry Telemanipulation*", IROS, Grenoble, France, September 1997.

[3] Fraunhofer IFF : "*Annual Report 1998: Interactive 3D training simulator using high level architecture*".

[4] Freedmann, P : "*Forestry Machine Simulators: looking for added value in trainings*", Canadian Woodlands Forum Annual Meeting, Montréal, Québec March 23-25,1998.
[5] Timberjack : "*The key to effective operator training*", latest brochure for Harvester simulators of Timberjack.
[6] Rossmann, J.; Wagner, T.: "*Forstmaschinenführer-Fortbildung in der dritten Dimension — ein Beispiel für die Zusammenarbeit von Ökologie und High-Tech bei der Waldarbeit*", Jahresbericht 1998 der Landesanstalt für Ökologie, Bodenordnung und Forsten (LÖBF), pp. 57-65, ISSN 1432-5926, Recklinghausen, 1999.
[7] Rossmann, J.; Wagner, T.: "*Forstmaschinenführer in der dritten Dimension*", Forst & Technik, Heft 2/99, BLV Verlag GmbH, München.

A00-37231

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

AIAA-2000-4169

THE DEVELOPMENT OF A MOTION SYSTEM FOR CABIN EMERGENCY EVACUATION TRAINERS.

Philippe Piatkiewitz*
Hydraudyne Systems & Engineering B.V.,
P.O. Box 32, 5280 AA, Boxtel, The Netherlands

Until recently Cabin Emergency Evacuation Trainers (CEETs) have been equipped with positioning systems to allow the training of aircraft evacuation in different crash positions. Now there is a growing demand for adding realistic dynamic motion to CEET's, since added realism will increase the confidence of Cabin Crews to handle aircaft emergencies and thus will have a positive effect on training. This paper gives an overview of the requirements for a CEET motion system which can act both as a positioning system and as a motion cueing system. A description is given of the development of such a motion system for large payloads in the range of 10,000 to 30,000 kg. The system is capable of simulating the dynamic aircraft behaviour during take-offs, (crash-)landings and severe turbulence. Since this new application of motion cueing is only just emerging, its actual effect on the quality of cabin crew training has yet to be investigated. First impression of its application however have been positively commented by both instructors and trainees.

Introduction

Improvement of the training of emergency procedures for aircraft cabin crews is an ongoing process, stimulated by world-wide regulations[3] on passenger safety and the demands of the airline's customers. The goal of cabin crew training is to provide the crew with *knowledge* of emergency procedures, which is taught in classroom sessions, and the *skills* to carry out those procedures, through practice in a realistic training environment, a Cabin Emergency Evacuation Trainer (CEET). A CEET is a mockup of the cabin of the real aircraft, which includes passenger and crew seats, galleys, aircraft doors, overhead bins and a cockpit. The purpose of a CEET is to replicate all aspects of the real aircraft relevant to the training.

In a training session, the instructor can choose from a set of basic flight scenarios (take-offs, (crash-)landings) using a graphical user interface in his instructor station which is located inside the cabin, within view of the trainees. To these basic scenarios the instructor can add a variety of different simulated emergencies and effects; like cabin fires, smoke, turbulence and cabin decompressions. Most training sessions will end with an evacuation of the aircraft in a certain crash position, depending on the scenario that was chosen. E.g. after a very hard landing the left main gear of the aircraft has collapsed such that the aircraft (and thus the simulator) comes to a stop at a certain roll angle. In this position the doors have to be opened and the slides have to be inflated, after which all passengers and crew have to leave the aircraft over the slides. After the evacuation is completed, the motion platform is lowered to it's settled position and a new training session can start.

Repeatedly training under realistic circumstances is known[5] to reduce the amount of stress the Cabin Crew experiences during a real emergency situation; since stress results from the unconscious comparison between the perceived required competence to handle the crisis situation and the internal image of one's competence. Thus, added realism in training contributes to the trainees confidence to handle the emergency situation.

One way to improve the realism in the training environment is to add realistic dynamic aircraft behavior to the CEET. Until very recently, CEETs where equipped with a positioning system to simu-

*System Engineer, Control & Servo Engineering, AIAA Member

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

late different crash positions of the aircraft, however, for the realistic simulation of in-flight and on-ground dynamic aircraft behavior, these systems are not suitable; a motion system capable of both the static positioning task and the dynamic motion cueing task is required.

The value of motion simulation for pilot training applications has been proven over the years. Correctly applied motion cues* provide the pilot with (time critical) information on the current state of the aircraft. As in the real aircraft, this helps the pilot with his task to control the aircraft.[2] Furthermore, motion cueing adds to the pilot's sense of realism, which has a positive effect on training.

For cabin crew training, the value of motion simulation has yet to be proven, but it is expected that it has a similar positive effect on training as in pilot trainers. Although the cabin crew has no control over the aircraft's state (unlike the pilot), in case of an emergency they have to make critical decisions based on the information they perceive. In a Cabin Emergency Evacuation Trainer this information is presented to the crew through:

- *visual cues*; via the cabin interior, various relevant (warning-)lights and indicators, other participants in the training (e.g. actors) and a visual display system which gives a representation of the cabin exterior.
- *auditive cues*; via a sound system, which mimics aircraft noises and auditive cabin signals as well as noises of passengers or cabin fires.
- *motion cues*; via a motion system
- *various sensory cues*; like smell, heat, air pressure.

Human perception is usually dominated by visual information, since in general we 'trust' our eyes more than any other sense. However, the information must be in view to be perceived and eyesight is easily inhibited, e.g. by darkness or smoke. In this case, situation awareness must be obtained from other cues, like motion.

In the following sections, the development of a CEET motion cueing system is described.

*a cue is a super-threshold stimulus that can cause reaction by the Central Nervous System, eliciting the perception process[1]

Requirements for a CEET motion-base

As seen in the previous section a CEET motion system must fulfill two main functions:

- *positioning*
- *motion cueing.*

To provide the positioning function it must be possible for the system to freeze the platform in an arbitrary pose†. In this pose the trainees will open the doors and evacuate over the emergency escape slides along the outside of the CEET. To guarantee the trainees' safety, all power must be disconnected from the motion system to prevent unexpected motion of the platform.

For the motion cueing task the system has to be capable of generating realistic motion cues from the viewpoint of the trainees. This means that the motion system must be able to generate accelerations with a high degree of accuracy and provide a large enough workspace to simulate acceleration onsets and sustained linear acceleration. (The workspace of a motion system is defined as the six-dimensional hyper-volume of translational and rotational motion capability of a pre-defined reference point attached the motion platform.) However, to limit the risk of injury to the cabin's occupants, the acceleration of the cabin should never exceed 2.5 g ($\approx$25 ms^{-2}). To ensure realism of the complete simulation, it is important that the platform motion is synchronized with the other cueing systems (audio and video).

The main technical requirements for a CEET motion system are:

- *payload*; to provide an adequate training environment the CEET cabin mockup will include all relevant sections of the aircraft. As a result of this, the payload for the motion system will have a large mass and large dimensions. The payload mass is in the range of 10,000 kg (e.g for a Boeing 737 aircraft) to 30,000 kg (e.g. for a Boeing 747). The dimensions of the payload are in the order of 13 to 20 meters in length and up to 6 meters in width (depending on the aircraft type).
- *settled height*; in settled pose all the motion systems actuators are fully retracted and the platform is at its lowest position. For a CEET

†A pose is a set of generalized coordinates describing the position and orientation of a body relative to an arbitrary axis system

this lowest platform position is important, since in a crash position where all the aircraft's gears have collapsed the cabin floor reaches a certain minimum height. The motion system should be able to reach this minimum height to ensure that the emergency escape slides are at the desired angle w.r.t. the floor.

- *excursions*; apart from providing sufficient excursions to allow realistic (dynamic) motion cues, the system must also be capable to reach several (static) crash-positions, e.g. when the nose-gear has collapsed the body of the aircraft will be pitched down over a certain angle.

- *motion envelope*; the motion envelope is the total volume required to accommodate the simulator. Under no circumstances it is allowed that the CEET interferes with other objects in its environment. If the available space in the facility is limited, the design of the motion system and the cabin must take this into account.

- *motion freeze*; it must be possible to freeze and shutdown the motion system (remove the hydraulic energy) in any desired crash-position to allow a safe use of the emergency escape slides in different crash positions.

- *nominal velocities and accelerations*; certain minimal velocities and accelerations are required to generate sufficiently realistic motion cues.

- *worst case accelerations and forces*; forces and the resulting accelerations must be kept below an acceptable maximum, such that the risk of injury to the passengers or damage to the equipment is minimized.

- *safety*; due to the nature of the usage of the system special attention has to be paid to the safety of operation of the system. In case of an emergency (e.g. power failure) the system must be lowered to settled position automatically within certain velocity limits. This requires the actuators to be velocity limited under emergency-stop conditions, but not under normal operation. To be able to practice evacuation of the aircraft in a simulator crash-position, the system must allow the doors to be opened in a platform pose other than settled. However if the system is in motion, opening an external cabin door must cause the system to automatically come to a stop in settled.

- *host-computer to motion system interface*; usually a host-computer calculates the necessary (aircraft-) motion and sends the results to the motion system's control computer over a network link. But, Since the flight scenarios are fixed, the required simulator motion files can be uploaded to the motion system's motion control computer in advance. This will alleviate the host-computer's task and reduce the amount of network traffic.

Apart from these main functional requirements, the motion system also has requirements for lifespan, mean-time between failures (MTBF), maintainability and costs.

Development of a CEET motion-base

Some of the functional requirements presented in the previous section are in conflict with other requirements to some extent and to arrive at an optimal motion system design a trade-off has to be made. One of the most profound challenges in finding an optimal design is due to the combination of requirements on payload, excursions and the limited height of the cabin floor in settled pose. To obtain sufficiently large excursions while not sacrificing settled height, the kinematic links which form the motion system (like linear actuators) must be placed under a very shallow angle with respect to the floor. With the large payload mass, this configuration can easily result in (unacceptably) high forces on the motion base and on the facility floor. Through an iterative process a motion base configuration which reaches the goals with respect to settled height, excursions and motion performance while minimizing the forces is saught for.

To facilitate this iteration process, special software tools have been developed to quickly evaluate the performance of any given motion system configuration. A set of functions have been written in Matlab[4] grouped into the Motion Designer Toolbox. With this toolbox a model of a parallel mechanism can be created from basic building blocks: motion platform, floor-pads, linear actuators, rods and cranks (hinges). Through a graphical user interface (see Figure 1) the parameters which define a motion base configuration (like joint locations, actuator stroke and rod-length) can be modified. To evaluate the quality of the design, various performance measures can be calculated such as excursion limits and workspace, nominal actuator velocities, acceler-

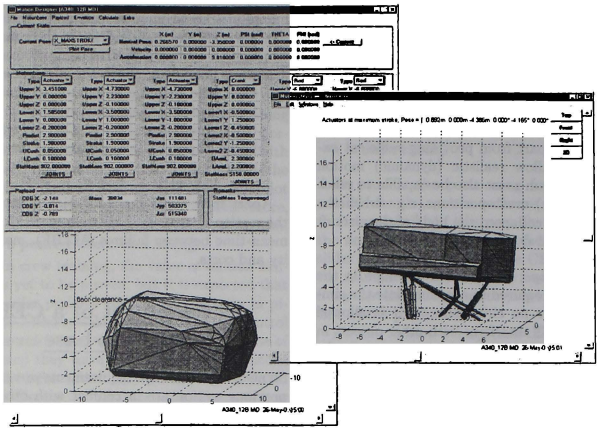


Fig. 1 Motion Designer User Interface. The figure shows the main parameter window (top) which allows easy editing of the motion system's design, a general layout view (right) and the result of a motion envelope calculation (bottom).

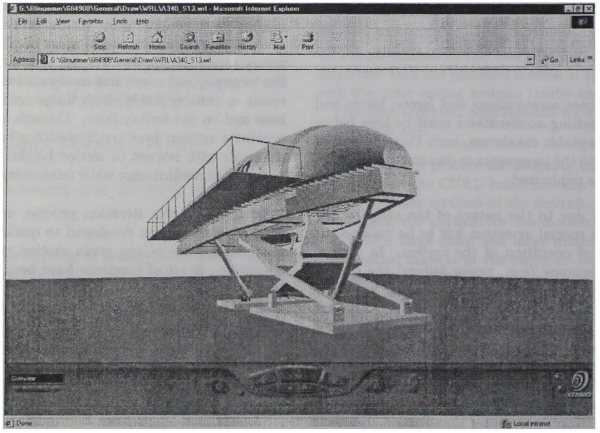


Fig. 2 Visualization of a motion design in a VRML-capable web-browser. The controls at the bottom of the screen can be used to navigate through the virtual environment.

ations and forces, worst case forces on the motion base and the facility, maximum joint angles and the motion envelope of the motion system and its payload.

For visualization, the motion base configuration can be exported to a VRML[‡]-file[7] including an animation of the platforms motion capabilities. Using a standard web-browser with a VRML plug-in (e.g. Cosmo Softwares Cosmo Player[8]) the motion system can be evaluated visually in a virtual environment. An example of a web-browser window showing a VRML model is shown in Figure 2. Using the mouse it is possible to "walk" through the design and do initial checks for leg-crossing or other types of interference. Also, presenting the design in Virtual Reality is particularly useful when communicating with other engineers or the customer.

If a suitable concept of a motion base configuration has been found the mechanical, hydraulical and electrical parts are specified and the design is released for detailed engineering.

Description of the CEET motion-base

Using the virtual prototyping tools described in the previous section, three types of motion systems have been developed and built, each for a different aircraft type (for a Boeing 737, Airbus A340 and Boeing 747 type aircraft). The difference between the three motion systems is mainly their payload mass, payload dimensions, settled height and required excursions. Table 1 shows the main capabilities of each type of motion system.

The CEET motion system is a parallel mechanism, consisting of a motion platform which supports the cabin and the external walkways, three hydraulic actuators, an A-frame and a scissor hinge. A photograph of an Airbus A340 CEET is shown in Figure 3. The motion system has three degrees of freedom: by extending or retracting the actuators independently, the platform can move in heave (up and down), roll (rotation about the longitudinal axis) and pitch (rotation about the lateral axis). The other degrees of freedom (sway, surge and yaw) are constrained by the A-frame and the hinge.

The hydraulic actuators are equipped with high performance servo-proportional valves, integrated position transducers and low-friction oil seals to guarantee smooth motion. For safety, each actuator has hydraulicly cushioned end-zones, which provide a hardware safety circuit to prevent damage to the system in case of a leg-runaway (e.g. caused by a broken cable connection). To make it possible to freeze the system in an arbitrary pose, each actuator is equipped with locking valves which close-off the oil volume at the actuator to prevent further motion.

The motion system is powered by a hydraulic power unit (HPU), which delivers approximately 100kW of hydraulic power. If more than one motion system is located in the same facility, it is possible to connect two or more pumps hydraulically. This way, each motion system can use the power of more than one pump, which makes is possible to simulate severe turbulence over a longer period of time. For added performance the hydraulic system is equipped with an accumulator station; this unit stores hydraulic energy in a large volume of nitrogen gas. It acts as a buffer in case the motion system temporarily requires more oil-flow than the hydraulic pump can deliver (e.g. during simulation of a landing bump).

The complete system is controlled by a PC-based real-time digital control system which includes motion cueing software. This Motion Computer receives information about the (simulated) aircraft state from a host computer over an Ethernet link. The aircraft state is represented by the specific forces[§], angular accelerations and angular rates of the aircrafts center of gravity. The host will also send data about the desired crash-position and information about simulator special effects like turbulence and landing bumps.

The motion cueing software transforms the aircraft motion to simulator motion via a motion cueing algorithm.[1] This algorithm generates optimal motion cues within the limitation of the workspace of the simulator. Taking into account the limited sensitivity of the human motion sensors, the aircraft motion is mapped onto the simulator workspace. For instance, sustained aircraft acceleration in surge (along the longitudinal axis) is transformed into a pitch angle of the cabin, such that one component of the gravitational acceleration is projected along the simulators x-axis. This gives the impression of a sustained horizontal acceleration. The rotation speed with which the cabin is rotated, however, has to be below the human sensory threshold. Although the motion system has no ability to move in surge (the A-frame prevents this) it is still possible to simulate sustained *acceleration* in surge.

[‡]VRML=Virtual Reality Modeling Language

[§]Specific force is the external force acting on a body per unit of mass, including gravitational forces.

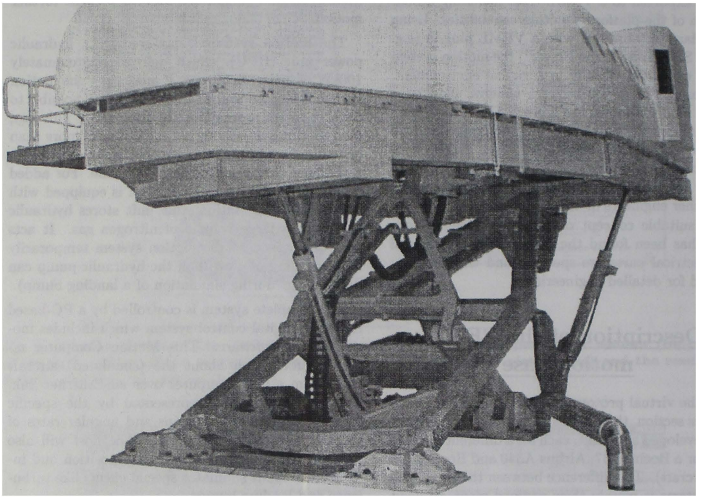


Fig. 3 A340 Cabin Emergency Evacuation Trainer.

Evaluating the effect of motion cueing in a CEET

The goal of adding motion cueing to Cabin Crew Training is to improve the quality of training through added realism. To evaluate the effect of motion cueing on training is not an easy task. In pilot trainers the effectiveness of motion cueing can be measured objectively (to some extent) by comparing the pilot's control actions in the simulator to his actions under similar conditions in the real aircraft.[6] In a CEET this method cannot be used, since the trainees do not generate control outputs which can be measured. However, the value of motion simulation could be established by comparing the respones of trainees to certain tasks in a CEET with motion cueing enabled to the responses of the same trainees in a CEET with motion cueing disabled (e.g. by comparing their heart-rates as a measure of added stress). The quality of training in a CEET could also be measured subjectively; e.g. by interviewing trainees and instructors about their experiences or visually comparing responses of cabin crews in the CEET to responses in the real aircraft.

At the time of writing of this paper, two motion systems have been in use for some months. The general impression of cabin crews and instructors is that motion cueing adds substantially to the trainee's sensation of realism. Now it is possible to train extinguishing a cabin fire in turbulence conditions without the instructor having to say to the trainee: "imagine we are experiencing heavy turbulence". This makes the instructor's task easier and the trainees task harder. However, since this application of motion cueing is only just emerging it is too early to draw conclusions on its effects on the quality of training. Further investigation is required.

Motion system type	HSE-3-MS-71-CET-2D	HSE-3-MS-79-CET-1D	HSE-3-MS-79-CET-3D
Aircraft type	B737	A340	B747
Motion system payload	10,000 kg	20,000 kg	30,000 kg
Platform length	13.5 m	14.2 m	18.1 m
Platform width	3.0 m	5.7 m	5.9 m
Height of cabin floor relative to ground level in settled pose	1.90 m	3.05 m	3.30 m
Non-Simultaneous Excursion Limits			
heave	-0.89..+1.12 m	-0.90..+0.79 m	-0.93..+0.93 m
pitch	-21..+15°	-17..+12°	-17..+12°
roll	-14..+14°	-20..+20°	-20..+20°
Nominal velocity			
heave	0.5 m/s	0.5 m/s	0.5 m/s
pitch	8 °/s	8 °/s	8 °/s
roll	8 °/s	8 °/s	8 °/s
Nominal Acceleration			
heave	4 m/s^2	4 m/s^2	4 m/s^2
pitch	35 $°/s^2$	35 $°/s^2$	35 $°/s^2$
roll	35 $°/s^2$	35 $°/s^2$	35 $°/s^2$
maximum acceleration in heave (landing bump)	1 g	1 g	1 g

Table 1 Main motion-system specifications.

As more experience with the use of dynamic motion systems for CEETs is gathered in the near future, it is likely that this type of highly realistic training environments will become the standard for cabin crew training.

References

[1]S.K. Advani. *The Kinematic Design of Flight Simulator Motion-Bases.* PhD thesis, Delft University of Technology, Dept. of Aerospace Engineering, 1998.

[2]R.J.A.W. Hosman. *Pilots peception and control of Aircraft Motion.* PhD thesis, Delft University of Technology, Dept. of Aerospace Engineering, 1996.

[3]Joint Aviation Authority (JAA). Jar-ops part 1: - commercial air transportation (aeroplanes), subpart o - cabin crew, 1998.

[4]The MathWorks. Matlab. version 4.2c, Natick, MA, 1997. http://www.mathworks.com/.

[5]National Transportation Safety Board (NTSB). Flight attendant training and performance during emergency situations. Technical Report SIR-92-02, NTSB, 1992.

[6]L.D. Reid and M.A. Nahon. Response of airline pilots to variations in flight simulator motion algorithms. *AIAA Journal of Aircraft*, 25(7):639–648, July 1988.

[7]Chris Marrin Rikk Carey, Gavin Bell. Iso/iec 14772-1:1997, virtual reality modeling language, (vrml97), 1997.

[8]Cosmo Software. Cosmo player. version 2.1, Islandia, NY, 1999. http://www.cai.com/cosmo/.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37232

AIAA-2000-4170

ATTAS Ground Based System Simulator - An Update -

S. Klaes
German Aerospace Center (DLR)
Institute of Flight Research
Braunschweig, Germany

Abstract

At the German Aerospace Center (DLR) a sophisticated in-flight simulator called ATTAS (Advanced Technologies Testing Aircraft System) is operated for research in the areas of flight-test, control-laws and hardware development.
As an important link in the chain of ATTAS flight test philosophy, the ATTAS ground based simulator plays an important role in the preflight development and testing of hard- and software in the loop, crew training and flight test preparation under real-time conditions.
Since every soft- and hardware-modification has to be fool-proof tested before being implemented into the real plane, the ground based simulator has to provide the flight-engineer with a research and test environment that matches the actual plane in all aspects from flight performance over handling aspects to all existing data flows.
Those demands for a realistic aircraft flight environment also result in a need for constant validation and optimization of the dynamic simulation fidelity.
This paper will highlight the hardware and software concepts of the ATTAS ground based simulator and will give a short overview on the methods used.

Introduction

ATTAS Inflight Simulator

The airborne testbed ATTAS (Advanced Technologies Testing Aircraft System) is a unique research tool with its in-flight simulation capabilities. Its potential of use ranges from complete in-flight simulation of different aircraft over research on new control technologies to investigations on man-machine interfaces and new cockpit designs.

Copyright ©2000 by S. Klaes. Published by the American Institute of Aeronautics & Astronautics, Inc. with permission.

Its key features are the real flight impressions of the simulated aircraft during the manoeuvres, including pilots viewpoint, environmental influences and accelerations.

The in-flight simulation is enabled by a sophisticated model following system that imposes the flight characteristics of the simulated aircraft onto the host aircraft. The principle of this strategy is shown in figure 1.

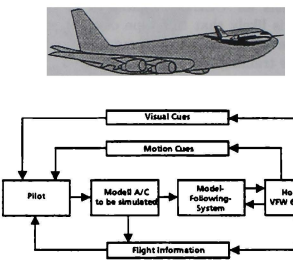


Fig. 1: In-flight simulation

The ATTAS in-flight simulator is based on a modified VFW 614 airframe that was completely remodeled to match the in-flight simulation requirements.

Next to the installation of a fly-by-wire steering, extended control computer system and the flight engineer stations, a whole new set of additional Direct Lift Control (DLC) Flaps was installed to increase aircraft dynamics and maneuverability.

On the right hand side the ATTAS in-flight simulator is still equipped with all basic control instruments, whereas on the left hand side a sophisticated, modifiable fly-by-wire control and instrumentation system allows for high flexibility in the different experiments to be flown.

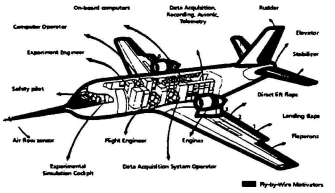


Fig. 2: ATTAS Modifications

To meet the high aviation safety standards the FBW system is designed as a duplex system; additionally the safety pilot on the right-hand side is at all times capable of transferring the controls back to his side, when advisable.
An extensive description of the ATTAS in-flight simulator is given in [1,2].

ATTAS System Simulator

One essential tool for the testing and development of new technologies with the ATTAS in-flight simulator is the ground based system simulator. It provides the research engineer with a complete development and testing environment for software evaluation and hardware-in-the-loop testing, which is the main focus of the ground-based system simulator. It creates a real-time simulation environment with the exact flightmechanical and environmental conditions applying to the real plane including all existing connections and data flows.

Once the experimental hard- or software is tested and verified to the specified requirements and satisfaction of the flight engineer, the generated code or instrument is transferred from the ground-based system simulator to the flying system with no additional changes, giving the mandatory safety for in-flight experiments [3].

Fig. 3 shows the interweavement of in-air and ground-based in-flight simulation:

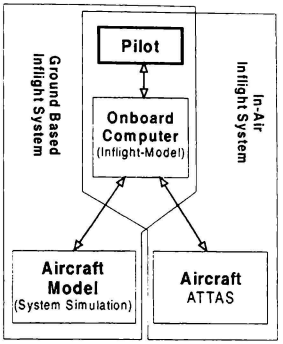


Fig. 3: In-flight Simulation

The system simulator also serves as a crew trainer for cockpit procedures and experiment handling, giving the pilots the necessary exercise and routine for safe in-flight experiments. Additionally to those key benefits new or extended control law concepts are developed and evaluated with pilot-in-the-loop experiments, studies on human-machine interfacing are performed and new display and visualization technologies are investigated.

All those features imply that it is crucial to offer the pilots and flight engineers a realistic cockpit and working environment on the ground. For that matter an original VFW 614 cockpit was acquired, mounted and outfitted. The cockpit controls are set up the same way as in the real plane with the FBW controls for the experimental pilot on the left-hand side and the original controls for the safety pilot on the right. The instrumentation for the experimental pilot matches those in the plane whereas the safety pilots instrumentation is replaced by programmable TFT-displays due to the need for easily interchangeable displays and extended data visualization for ground tests.
Additionally a generic sound simulation for basic cockpit sounds and background noise and a, at the moment, one channel video projection system are installed to increase the

realistic impression. A cockpit view of the ground based system simulation is depicted in figure 4:

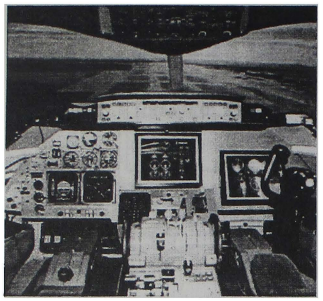


Fig. 4: Ground Based Simulation - Cockpit View

The whole setup can be controlled and monitored by an extensive supervisor station where different scenarios can be generated, specific experiments are loaded and every single data flow or flight state can be visualized and checked.

Hardware Realization

To obtain the task of creating an exact duplicate of the flying system, a sophisticated network of computer systems had to be realized.
As the heart of the system functions the VME-based ATTAS interface computer (AIC), which manages all data flows by collecting, transforming and distributing data to the applicable system simulation computers and all other peripheral devices.
Since such a simulation incorporates a lot of separated sub-systems and requires a transfer of large amounts of data, the communication between the AIC and the main components is realized by the use of reflective memories (RM), allowing for high data transfer rates.

For the simulation of the basic airplane and its environment a special high performance simulation computer (RTS) is used, guaranteeing a realistic representation of the flight dynamics and all outside environmental influences. It is based on a power PC system using a sophisticated real-time programming language (ADSIM) providing the necessary performance [4].

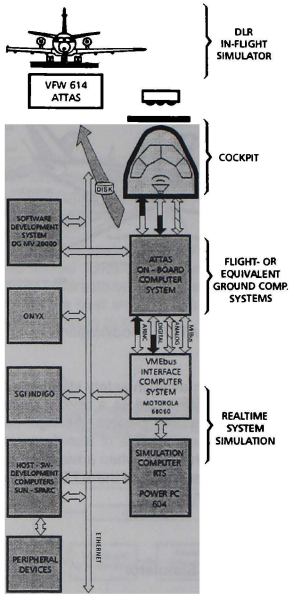


Fig. 5: ATTAS system simulation

By generating the necessary flightmechanical and sensory data in the required data formats, this connection of real-time simulation system and interface computer provides the necessary environment for the on-board computer system, in which the in-flight experiments are realized. This ground-based on-board computer system is set up in the same way as in the real plane consisting of a ring of five flight-computers connected by an optical fibre, handling the necessary flight computation and the different in-flight experiments.

Next to the core simulation setup, a variety of peripheral devices are connected to the

ground-based simulation by ethernet links via the AIC.

For the optical impressions a one channel video projection system powered by a SGI Onyx infinite reality engine is supplied by the HIC with all visual data.
Generated by a SGI Octane computer are all instruments and information units on the right-hand safety pilot side, as well as outside at the control station.
To push the reality level a little further, a PC based 8-channel sound simulation system is also implemented to the whole setup, providing engine noise and call-outs.

Finally the ground-based system simulator is equipped with the same extensive data recording unit as the in-flight simulator, what is required for post-flight analysis as well as for control and validation aspects.

Software Realization

The development and design of new control software and in-flight experiments on the ground before going into flight, requires a high bandwidth and detailed flight mechanical model. The model complexity has to be very elaborate to give a realistic representation of the ATTAS full flight envelope and to ensure a 1-to-1 transferability of experimental software to the flying system.
This conditions that in the system simulation not just the equations of motion and flight-mechanical derivatives have to be calculated but every mechanical subsystem in all aspects as well.
To cope with this task a generic, nonlinear, physical based model of the flying system including all subsystems was developed with respect to the real-time conditions. This model was then transferred to the modular macro programming language ADSIM running on the RTS, allowing a vector and matrix formulation of the modeled phenomena [5].

By doing so, a clear structuring of the model was achieved, with the possibility of adding model improvements and/or modifications more easily.

The generated ADSIM code is loaded directly into the RTS with no additional operating system involved allowing high performance computing of up to 500 calculation frames per second.

Such a high performance is necessary due to the complexity of the aircraft actuator systems, which would otherwise generate numerical instabilities, resulting in simulation inaccuracy [6].

Due to the large amount of data to be transferred and transformed between the aircraft simulation and on-board computer systems, an extensive interface software was developed. Every time the RTS calculates a new simulation frame a whole set of flightstate data is exchanged via the VME-bus based interface computer AIC, who converts every datum into the necessary format (IEEE, ARINC, etc.) and distributes and collects them to/from the above mentioned on-board computer system, simulation computer and peripheral devices.
In order to keep up real-time conditions while transferring data between all the computer systems, a great effort is done with the scheduling of all the tasks.

The interface computer system runs with the pSos real-time operating system using C-code as programming language.

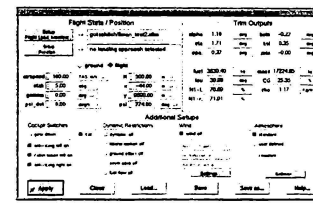


Fig. 6: Simulation Control Sample Screen

Data interchange between the time critical components is done via RM, whereas all other devices, like sound, visual simulation and displays and data recording, are connected to the AIC via a common ethernet connection.

The whole setup can be controlled and monitored with an elaborate control program running on a SUN workstation. With it, the simulation environment, such as position and flightstate, as well as environmental conditions like wind and visibility can be set up and controlled (see Fig. 6).

At the moment three airports (Frankfurt, Hannover and Braunschweig) can be chosen from, including all necessary navigational aids. A screenshot of the position generation via navigational maps is depicted in figure 7.

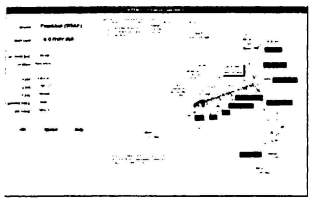


Fig. 7: Position Generation

Additionally, as mentioned prior, every data flow within the simulation can be selected, visualized and monitored as required. This is necessary for the supervisor on the control station as well as for the pilot inside the cockpit. It gives the ability of directly monitoring the simulation status and controlling reactions to specific inputs online and in real-time.
For this purpose a software tool called VAPS is used to generate control displays and data plots (see fig. 8).

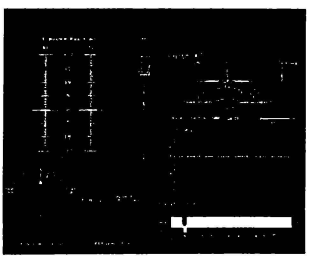


Fig. 8: Quicklook Display

Validation and Optimization

Such a highly specified simulation of course entails a constant process of validation and optimization due to structural and mechanical changes in the real world. The real-time simulation model has to be checked frequently against the real system to guarantee model accuracy for all incorporated systems.

For an operational validation of time-independent models, the accuracy of the model output is critical, whereas for time-dependent applications like this one, the quality of the characteristic system behavior during the crucial time intervals is of higher interest [7,8].

Up to now, a set of various control inputs (doublets, 3-2-1-1) was given as an input to specific systems of the plane as well as the ground-based system simulator. The resulting answer of both systems was then compared and the simulation validity was determined accordingly. In case of model mismatches, the system simulation had to be updated and optimized according to the new situation.

To determine the model quality was a sometimes very time consuming process since many specification points had to be defined to generate the tolerance function.

Now a slightly different approach is taken that promises good results in determining model validity, both in precision and time used. This approach focuses on time patterns rather than individual data points and evaluates model validity, returning parameters to facilitate the following optimization process.

Starting from the real world behavior as a reference, a set of one or more weighted tolerance functions is defined, based on specifications, expert knowledge and a priori experience. With those functions, areas of validity for every point of the system behavior are constituted. By merging all those 'tolerance areas' over the complete time range, the so called 'tolerance corridor (TC)' is generated comprising the complete model behavior including all acceptable tolerances (see Fig. 9).

By comparing this tolerance corridor to the response of the simulated system, for every point of the time range

$$q = t_q, (f(t_q))$$

the percentage of the model behavior within the boundaries of the corridor is determined with the 'scoring function' [9]:

$$\phi_g(g,f) = \frac{100}{N}\sum_{i=1}^{N}\phi_g(t_i, f(t_i))$$

with

$$\phi_g(Q) = \begin{cases} 0 & if \quad q \in TC_g \\ 1 & if \quad q \notin TC_g \end{cases}$$

The value of the scoring function is used as an indicator of the model quality, ranging from 0 to 100 percent.

This scoring function is then used as part of an optimization process based on an inductive learning algorithm. It determines the quality of the simulated model and returns this value back to the process. Based on this value and knowledge based information, the model parameters are modified to increase model accuracy. The process is repeated until the predefined precision criteria is reached and the model answer matches the tolerance corridor accordingly.

The main advantage of this approach, next to the high level of automation, lies in the possibility to validate and optimize multi-parameter models. Once the tolerance corridor is generated, the whole process runs autonomous and returns the optimized model parameters as an output.

At DLR this approach is used online in combination with the system simulation. The system to be validated and optimized is triggered with an input and the resulting answer is then evaluated. In the next step the predefined model parameters are modified, fed to the model and a new evaluation starts until an optimum set is found.

The whole process runs automatically and shows good results concerning economy of time and simplicity.
A schematic of this process is shown in figure 10.

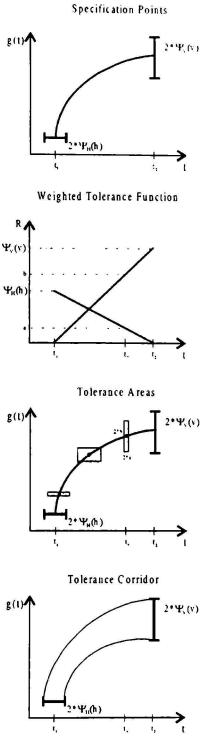


Fig. 9: Tolerance Corridor Generation

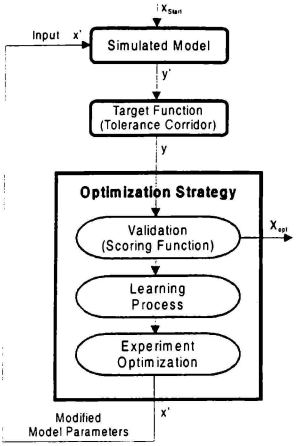


Fig. 10: Optimization Process

Outlook

As the computer science improves, the demand for faster, more accurate and more realistic simulation scenarios occurs. For that matter several upgrades of the systems, the flying as well as the real, are planned. For the ground based simulation as well as for the aircraft a new experiment computer system based on a power PC system will be added to enable more extensive in-flight simulations and programming and handling comfort.

For the ground based simulation an improved 2-channel visual system will be installed along with an expanded sound generation system to enhance simulation reality. Additionally a constant renewal and upgrade of hardware components is in progress to raise the reliability and speed of the system.

As predicted for the future, more and more research areas will and can be supported or even be covered by modern simulation techniques. They provide accurate and time and cost effective research environments for the engineer and gain importance every day.

For that matter the ground based in-flight simulation facility at DLR will expand its activities beyond the regular in-flight system simulation. For the future it is the intention to spread the simulation capability and to place different generic aircraft or systems for research purposes at the disposal of the research engineer next to the key feature in-flight system simulation.

Conclusion

This paper gives an overview of the structure of the in-flight simulation facilities at DLR. The principle of in-flight simulation is introduced in short along with the necessary hardware modifications of the aircraft.

Crucial for successful in-flight simulation is a sophisticated ground-based system simulation that enables development, research and training prior to take off.
The structural setup of this system simulator for ATTAS is described, highlighting the interfacing and real-time simulation of the aircraft stressing that such a highly specialized system demands the use of state-of-the-art technology in order to provide satisfying research results.

Emphasizing the need for highly accurate models under real-time conditions the utilized software and tools are presented and a new approach for validation and optimization of these models is introduced.

In conclusion DLR disposes of a highly developed ground-based system simulation as a basis and support of the in-flight simulator ATTAS. It is highly adequate for the challenging software and hardware-in-the-loop preflight tasks it is designated for and will be constantly updated and upgraded to match the requirements of a modern simulation facility.

References

[1] Hanke, D.: ATTAS - DLR's Flying Testbed for Aeronautical Research..
Lecture Series at the National Aeronautical Laboratory (NAL), Bangalore, India, 30 November - 08 December 1992, (1992)

[2] Bauschat, J.-M.: Die Technik der fliegenden Simulation am Beispiel des Flugversuchstraegers ATTAS der DLR..
Seminar "Flugsimulation/Flugmeßtechnik", Berlin, 01. Dezember 1994, (1994)

[3] Saager, P.: Real-Time Hardware-in-the-Loop Simulation for 'ATTAS' and 'ATTHeS' Advanced
Technology Flight Test Vehicles..
AGARD Guidance and Control Panel, 50th Symposium, 22-25 May 1990, Izmir, Turkey., AGARD-CP

[4] Saager, P.: The Use of RTS with CE-2 for ATTAS Ground Based HIL-Simulation..
ADIUS Europe 94, Nürnberg, 27. - 28. April 1994, (1994)

[5] Heutger, Bauschat, J.-M.: Entwicklung eines nichtlinearen Flugzeugsimulations-Programms für die In-flight Simulation mit dem Flugversuchstraeger VFW 614 ATTAS

[6] Howe, R.M.: Dynamics of Real-Time Digital Simulation
Applied Dynamics International,
Ann Arbor, Michigan, 1986

[7] Bolci, G.: guidelines for Successful Simulation Studies,
Prodeedings of 1990 Winter Simulation Conference, 1990

[8] Comfort, J.C.: Simulation Model Testing,
Proceedings of the 20th Annual Simulation Symposium, Tampa, FL, 1987

[9] Barton, R.: Maschinelles Lernen zur Validation von Simulationsmodellen,
PhD-Report, Institute of Flight Research, Braunschweig, 1998

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37233

AIAA-2000-4173

THE PILATUS ENGINEERING FLIGHT SIMULATOR

Abstract

During the past two years the Aerodynamics and Flight Simulation office of Pilatus Aircraft Ltd. has developed an Engineering Flight Simulator (EFS).

The original idea of a tool running on the desktop computers of the office's engineers to check the flight mechanics of an airplane developed into a proper real time, "pilot in the loop" facility. It is based on a tandem cockpit (derived from the Pilatus PC-9 trainer aircraft), with an electromechanical control loading system, a large screen and a high-fidelity TV projector for imaging. All hardware is installed in a dedicated room, with a separate control station for the engineer. Four computers power the whole system: three PCs and a Silicon Graphics workstation for the image generation. The flight data are presented to the pilot through a simulated HUD that reproduces faithfully the system installed on the PC-9.

The EFS is being used to aid development of new products and to study improvements to existing airplanes. It allows the engineers to simulate the flight mechanics of the vehicle both in normal flight and in extreme attitudes with the pilot in the loop. In the near future the EFS will be equipped with real avionics and instrumentation and it will be used also for the study of the human-machine interface.

The aerodynamic database and the flight mechanical model of the airplanes are developed within the Aerodynamics and Flight Simulation office. Tests have been conducted both in a rotary-balance and forced oscillation wind tunnel facility and in a static, high Reynolds number wind tunnel facility to gather aerodynamic data for a Pilatus trainer airplane extending from –90 to +90 degrees of angle of attack. The dynamic tests conducted with the rotary balance and forced oscillation rig allow a good simulation of the aircraft behavior in the post stall region and in the spin. Stall, spin entry, spin stabilization and recovery can be predicted; the dynamics of the airplane and the effects of the controls can be simulated leading to a higher fidelity vehicle dynamics.

The ability of the EFS to simulate correctly an airplane has been proved: the aeromodel built for an existing aircraft was validated on the basis of flight test data, both in the normal flight regime and in the spin. This allowed us to gain confidence on the method and assumptions made to build a correct aeromodel.

The EFS will be used in the future to conduct part of the certification tests of new aircraft, enhancing safety, cutting down flight test hours, costs and time to market.

The paper gives a brief description of the hardware and software of the Pilatus EFS and then focuses on the process of partially replacing the in-flight research and development through simulation and on the verification and validation of an aeromodel.

Why an Engineering Flight Simulator ?

During the design of an airplane, after the layout is completed and the basic systems are defined, it is useful to have a tool to test the solutions adopted. The CAD systems allow to virtually "assembling" the airplane during the design phase checking the procedures, the dimensions and preparing the tools. An equivalent system is needed to test virtually the operativity of the airplane and its performance.

A tool that allows the evaluation of the flight characteristics is extremely useful in order to predict the behavior of the airplane and the fulfillment of the design requests. Widely used are the methods to predict analytically the stability of the aircraft but these, although useful, don't give to the pilots and engineers a tangible, immediate idea of what the aircraft will do during flight and, most of all, don't allow the pilots to "fly" it.

The cost of the development of a new aircraft reflects, among others, the years needed for its design, prototyping, flight-testing and certification. One of the most important aspects of this process is the time-to-market, the period from the start of the project to the first delivery.

An early evaluation of the airplane flight mechanics gives the pilot the chance to suggest some improvements in terms of flight characteristics. The premature discoveries of possible deficiencies in the flight envelope can save valuable time during the design phase of the aircraft and, at the end, reduce the time-to-market.

The tool that permits this process is the Engineering Flight Simulator (EFS), which can be thought of as a testbed, in continuous development in parallel or ahead of the real airplane. An EFS can fulfill other tasks such as the study of the Human Machine Interface (HMI), the ergonomy, the linkage between the onboard computers and electronic devices, the testing of enhanced stability or automatic flight systems and the calculation of structural loads during peculiar maneuvers.

Nowadays a training flight simulator is a "must" for every new aircraft. For this scope a previously developed EFS can be able to provide a very detailed aerodynamic database and flight mechanics model plus all the systems and avionics simulation that were already implemented in the EFS.

Copyright ©2000 by Lucas Marchesini & Pilatus Aircraft Ltd.. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

The Pilatus Aircraft's Experience



Fig.1 The front of the cockpit and part of the screen.

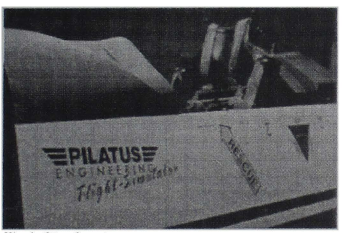


Fig.2 The front seat.

During the past couple of years in Pilatus Aircraft Ltd. an Engineering Flight Simulator has been developed. The first idea was to provide the Aerodynamics and Flight Mechanics office with a software tool that would enable the engineers to analyze the flight behavior of the airplanes in response to pre-recorded control inputs, i.e. time histories of controls deflections, deriving from flight testing. The aeromodels, composed of the aerodynamic databases and the flight mechanics models, would have been realized in house using the aerodynamic data calculated with theoretical or semi-empirical methods or gathered during wind tunnel and flight tests campaigns. It was decided that the "core" engine solver of the six degrees of freedom equations of movement would be off-the-shelf software: D-Six®, from Bihrle Applied Research. The main reason of going for an off-the-shelf product rather than realizing the program internally was the efficiency of human resources usage. D-Six® is like a "platform" on which all the aeromodels prepared by the engineers can run in real time and, apart solving the six-degrees-of-freedom equations, it offers utility functions such as displaying and plotting the simulation data, manipulating coefficients curves, matching pre-recorded flight test data ("overdrive" function).

To take advantage of the possibility of "flying" the virtual aircraft with a computer game device (joystick) it was decided to create a connection to an image generator which would provide the pilot with an out-of-the-window view on a 20" monitor.

The following natural step was to create a more realistic environment for the pilot. This led to the idea of setting up a real airplane cockpit with flight controls and forces, main systems controls and a big flat screen, in a dedicated room. A separate room was intended to host the control station for the EFS operator and a facility for the flight test engineer.

It was decided not to go for a moving base simulator because the cost of such a system is high compared to the benefit it gives, especially when fully aerobatic airplanes have to be simulated. In fact the longer the accelerations to be simulated last (typical during aerobatic maneuvers), the less realistic a moving base system is.

Since the EFS development was connected to the development of the new Pilatus' trainer, it was decided to use it also as a testbed for the avionics, which are going to be linked to the rest of the simulator in a short time and will allow some pilot's evaluation of the equipment and the HMI.

The keywords at the base of the EFS are "flexibility" and "reconfigurability" that allow this tool to be quickly setup to test in advance every concept for the aircraft.

Hardware and Software description

The Pilatus Aircraft's EFS has the external appearance of a tandem PC-9 turboprop trainer cockpit, rebuilt from the firewall to the rear seat frame using, as much as possible, dismissed parts to keep the cost as low as possible.

It is powered at the moment by four computers, linked in a mini Ethernet network, namely:

- the host simulator computer;
- the image generator workstation;
- the control loading system computer;
- the development computer in support of the control loading system.

The Host computer

The host is a Pentium III PC based computer, powered by Windows 95®, which runs the D-Six® software.
D-Six® is a solver of the six-degrees-of-freedom flight equations which presents a partially reconfigurable Windows interface featuring a plotting routine, useful for the engineering analysis, to trace graphically the behavior of whatever variable involved in the simulation.

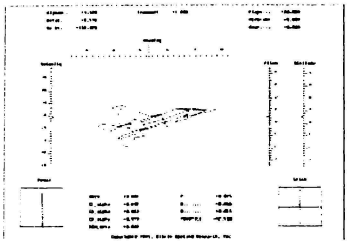


Fig.3 The D-Six® simulation window.

The aeromodel is composed of the aerodynamic database and the flight mechanics model. The first one is assembled in D-Six®, under a certain project name, using the tables produced from the data gathered during wind tunnel and/or flight test campaigns or calculated theoretically or with semi-empirical methods.
The second one is realized with a code in C++ that generates a Windows dynamic link library (DLL). To "fly" an aeromodel D-Six® loads the project and the correspondent DLL.
The C++ code allows the user to generate, specify the functionality and control whatever simple or complex system of the aircraft. In this part of the system is also where the secondary flight controls (gear, flaps, airbrake, fuel system, etc.) present in the cockpit are assigned to functions that describe the airplane systems.

The Control System

Original airplane mechanical components or precisely milled parts were adopted for the cockpit control linkage, where low friction and minimal free play are an issue. Behind the rear seat frame it is placed the control loading system, which is entirely mounted on a trolley that can disengaged from the cockpit stand. Onboard this trolley there are the three electric motors that generate the forces on the controls, their power suppliers and the amplifiers which process the signals coming from the Control Loading Computer (CLC), also placed on the trolley. Fokker Control Systems provided these components. The connections between the sticks and pedals and the electric motors were realized using low friction and high precision machined parts to minimize the uncorrectable free play and histeresis.
Thanks to the solution of the detachable trolley, the whole Control Loading System (CLS) is quickly removable and refittable to other cockpits that have the same connectors, requiring only the change to the appropriate software model.

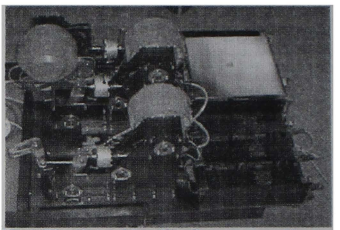


Fig.4 The Control Loading System assembly on the trolley, fixed to the cockpit stand: the 3 motors, their power suppliers and the CLC are visible.

The CLC is a Pentium based machine powered by the real-time operative system VX-Works, with two datacards to communicate with the amplifiers. It executes the engines' control loop at 5000 Hz while calculates the user's control model at 2500 Hz.
The user's models of the control system are aircraft dependent and have been realized in-house, reflecting accurately the behavior of the real systems of the Pilatus' simulated airplanes. Apart from simulating the aerodynamic dependent forces (deriving from the hinge moments on the control surfaces) on the controls, it takes into account the friction, the stretching, the free play and the inertia of every component of the control system of the airplane.
An interesting feature of the CLS is the possibility to modify all parameters in real-time, which allows a work in close cooperation with the pilots, whose movements and efforts on the controls can be traced accurately.
The CLC calculates the forces acting on the controls based on the data received from the host simulator computer, like the aerodynamic pressure and the local angles of attack, to which is linked via an UDP protocol. It is as well linked to another PC, which serves as a controller and development machine for the system. On this one the operator can see a graphical representation of the system, with all the components, the values of the parameters used by the control system model. From this station it is possible to monitor the movements of the controls and the forces exerted by the pilot and modify the values of some parameters without stopping the simulation. The models of the control system are written in C

language and to modify and recompile them the simulation must be stopped to allow the loading of the compiled code on the CLC.
The flexibility of the system allows changing the control system model loaded in few seconds hence giving the possibility to change quickly the airplane flown.

The Image Generator

The generation of the out-of-the-window view is performed by a Silicon Graphics Octane workstation powered by two R10000 processors, with 512 Mb of RAM and two SI graphic heads, one of which with 4 Mb of texture memory. This video channel sends the signal to the projector placed above the cockpit while the other channel was intended to be used to generate a virtual instrument panel on a monitor in front of the pilot.



Fig.5 The cockpit and the CRT projector.

The software to display the out-of-the-window view has been realized in house and is based on Performer®. Two terrain databases are available: one with a low level of detail and an area of 100x100 km, used for high altitude flight, and a smaller, more detailed one, used for low level flight.
The Octane with the SI graphic head is fast enough to display these terrain databases at a frame rate of 30 Hz in almost all flight conditions. The frame rate decreases sensibly during rapid roll maneuvers at high altitude, when many polygons of the database pass quickly in the displayed view angle.
Considering that the EFS is used to evaluate the flight characteristics of the aircraft, it has been decided not to invest resources for a big and detailed terrain database. Sometimes the pilots fly in the "deep blue" without even seeing the terrain, concentrated on the Head Up Display (HUD) and the data displayed. The HUD is projected in the same channel of the out-of-the-window view and is superimposed on the landscape view, with the dimensions corresponding to the angle under which the pilot sees it in the aircraft. The symbols are the same as displayed in the standard Pilatus PC-9 HUD but can easily be changed to test new graphic arrangements or different data displayed.
As the EFS has no instrumentation, the HUD, apart from some indications in the bottom corners of the screen, is the only mean to provide the pilot with the flight data. To date it proved itself sufficient to transmit to the pilots all the data necessary for the flight mechanics evaluation of a fully aerobatic trainer aircraft. The solution is economic, effective and doesn't present the problem of collimating the image.

The Aeromodels Development

The term aeromodel designates the group of aerodynamic tables which describe the effects of the change of the value of some parameters (i.e. angle of attack, sideslip angle, roll rate, etc) on the aerodynamic coefficients, and the flight mechanics model, which contains the information on how to assemble the tables.
At the time being the Pilatus' EFS can simulate four different airplanes loading the correspondent aeromodels: the PC-9, the PC-12, the P.o.C. (a modified PC-7MkII) and the PC-21.
The PC-9 dataset is being used in the flight training devices for this airplane. The PC-12 dataset is used in a Level 6 FAA certified flight training device.
The following graphs show how much the dimensions of the databases have increased stepping from one airplane to the next one, which testifies the growing complexity and detail of the models. The data are organized in 2D rows and columns text tables; 3D and 4D data structures are realized grouping together 2D tables.

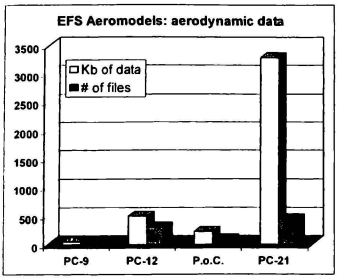


Fig.6 Aerodynamic data of the aeromodels.

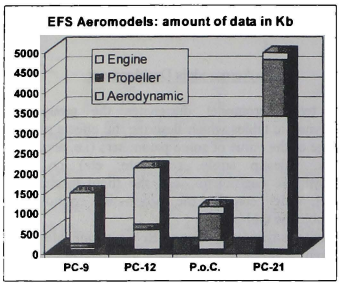


Fig.7 Total amount of data of the aeromodels.

The Aeromodels Validation

The validation of an aeromodel is the comparison of how the simulated aircraft and the real one fly. It is important to be aware of the differences between them to use the EFS as a proper tool, respecting its limitations.

At the base of using and validating an aeromodel there is the understanding that the simulator calculates the forces acting on an airplane, and then its movements, hence the physical relationship cause-effect has to be proved. In other words, the laws of physics of the airplane motion drive the real flight while simulation calculates the physics. Hence the last one allows knowing directly, not inversely, and instantaneously the quantities that determine the movement of an aircraft (i.e. all the aerodynamic coefficients, forces, accelerations).

Hence to validate an aeromodel some data that are not usually recorded during the normal flight testing procedures are required from the flight test departments. This is the reason why it's well known that the certification of a simulator or a flight-training device is, in a certain way, more complex than that of the real aircraft that it represents.

The validation of an aeromodel is a delicate operation that requires

- the existence of the real aircraft, which looks obvious, but means that the aeromodel used to develop a new aircraft prior its production has to "fly" without being validated;
  specific data gathered during flight test campaigns;
- a pilot able to reproduce in the simulator some maneuvers performed in flight;
- for the flight characteristics, which are more "subjective", a pilot able to distinguish the influence of some cues he cannot get in the EFS from the effect of the flight mechanics calculated.

It must be remembered also that the validating data refer to one airplane or to a limited number of them (the test airplanes) which are in a certain status (number of flight hours, maintenance, etc.) but must be representative of all the aircraft of that model.

The Validation of the P.o.C. model

Although the amount of data available for the aeromodel of the P.o.C. is not very large and the quality is not optimal for the build up of an aerodynamic database able to represent the whole flight envelope of the aircraft, it was decided to proceed with the validation. This was done to gain experience and establish a clear procedure for the aeromodel that had to come, the one of the PC-21.

The P.o.C. is a PC-7MkII modified in the wing, the tail and the engine. The aircraft is fully instrumented and regularly used by the flight test department. This fact granted the availability of lots of data needed for validation purposes.

During the validation process it was experienced the fact that the hardest model to get working correctly was that of the control system. The reason is thought to be in the many parameters (hinge moment coefficients, frictions, free plays, etc.) whose values are difficult to be determined exactly.

The validation of the aerodynamic database instead was easier.

In addition to the condition of airplane trimmed for straight flight at different airspeeds and power settings, the validation was conducted analyzing some maneuvers:

- stick force per G;
- rapid rolls;
- sideslips;
- looping;
- spin.

Here follow the plots of some results from the EFS data (hollow marks) compared to the flight test.

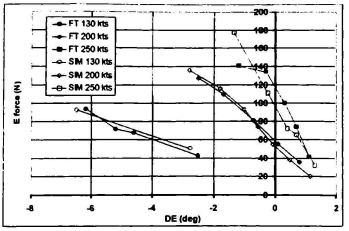


Fig.8 Stick elevator force vs. elevator deflection.

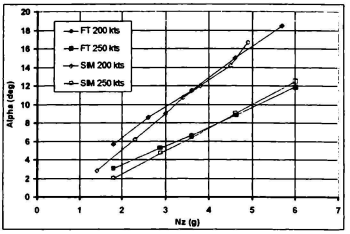


Fig.9 Angle of attack vs. load factor.

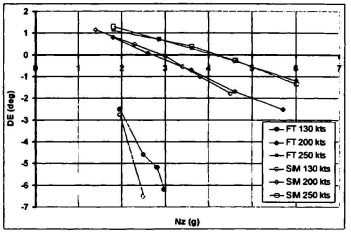


Fig.10 Elevator deflection vs. load factor.

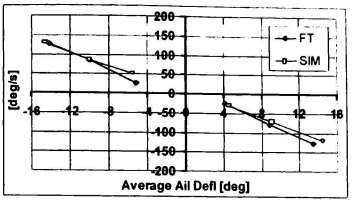


Fig.11 Roll rate vs. average ailerons deflection at 150 kts.

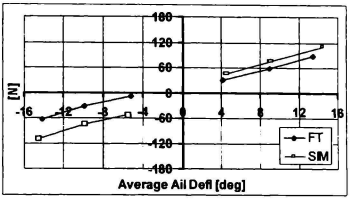


Fig.12 Ailerons stick force vs. average ailerons deflection at 150 kts.

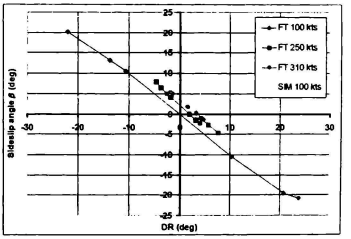


Fig.13 Sideslip angle vs. rudder deflection

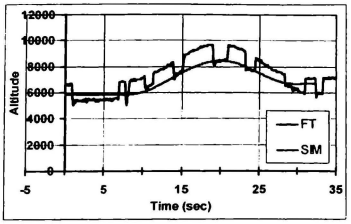


Fig. 14 Looping: altitude vs. time.

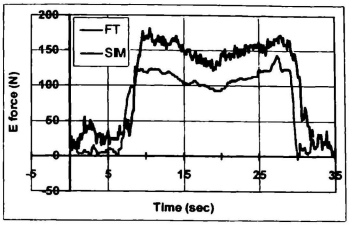


Fig.18 Looping: stick elevator force vs. time

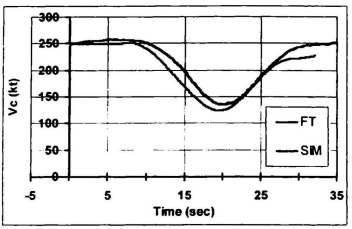


Fig. 15 Looping: airspeed vs. time.

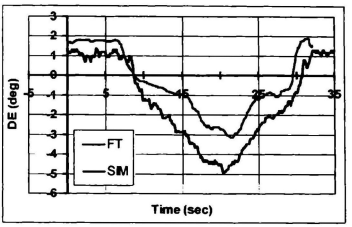


Fig.19 Looping: elevator deflection vs. time.

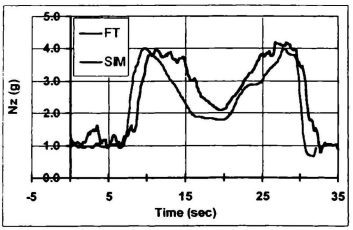


Fig. 16 Looping: load factor vs. time.

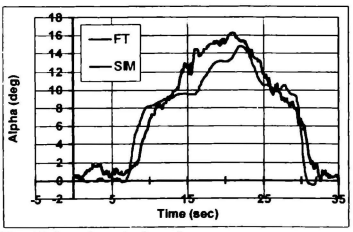


Fig. 17 Looping: AoA vs time.

The Performance of the EFS

The ability of the EFS to represent the behavior of an aircraft realistically is greatly dependent on the quality of the data available.

The P.o.C. aeromodel was built using mainly the data gathered in a vertical tunnel equipped with a rotary balance, in a range between 0° and 90° angle of attack (AoA). The P.o.C. is hence able to realistically simulate the stall, the post stall behavior, the spin and the recovery from the spin. Because the Reynolds number (Re) of the model in the wind tunnel was low, this aeromodel is representative of the real aircraft only in a reduced flight envelope.

With the following aeromodel, the PC-21, it has been possible to build a very detailed aerodynamic database with the data gathered both in the vertical tunnel with the rotary balance and in a bigger wind tunnel that allowed higher Reynolds numbers.

For this model the data measured in the vertical tunnel range from -90° to +90° AoA, while the data collected in the higher Re wind tunnel ranges from -30° to +30° AoA. It was necessary to integrate the data coming from the two wind tunnels. The procedure has been to extend the high Re data from -30° to -90° AoA and from +30° to -90° AoA using the data from the vertical tunnel. Thanks to the rotary balance installed in this facility the dynamic rotary data were measured. The implementation of these

data in the aeromodel allows the simulation of the stall and post stall behavior, of the spin and the spin recovery. The spin entry is a delicate point because it is not a steady condition, while the dynamic data refer to steady conditions.
An interesting possibility of the EFS is to allow the simulation of malfunctions of some systems. For example the P.o.C. and PC-21 aeromodels allow the study of the behavior of the aircraft with the spoilers locked open or closed. This has been used to verify the capacity of the pilot in facing such an emergency, checking the forces on the controls needed to keep the airplane in acceptable flight conditions. The aeromodel of the PC-12 allows a non-symmetric flap retraction to check what are the best actions the pilot has to take when this event happens.

Conclusions

In a relatively short time, two years, the Aerodynamics and Flight Simulation office of Pilatus Aircraft Ltd. has set up an engineering flight simulator with all the features necessary to make research and development work on the flight mechanics characteristics of the airplanes.
The EFS is being currently used for the development of a new aircraft investigating the behavior of the plane throughout the flight envelope. It is planned to use it also during the certification phase of the airplane to perform part of the flight test activity, contributing in reducing the time-to-market of the aircraft.
A tool like the EFS must be developed and improved continuously to keep it ahead of the real aircraft in order to test in advance new concepts and modifications.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37234

AIAA-2000-4174

A Description of the Cockpit Motion Facility and the Research Flight Deck Simulator

R. Marshall Smith*
NASA Langley Research
24 West Taylor Road
Hampton, VA 23681

Abstract

The NASA Langley Research Center's (LaRC) Visual Motion Simulator (VMS) has provided six degree-of-freedom motion cues for research programs since 1974. During that period the VMS has provided a generic cockpit with motion performance for a large number of critical research programs. The cockpit has been continually upgraded, however the age of the simulator, coupled with future requirements that the VMS cannot meet, has necessitated the development of the Cockpit Motion Facility (CMF). Some of these requirements include greatly improved motion characteristics (acceleration, maximum excursion, bandwidth), full-mission capable simulators operating in either fixed-base or motion-base modes, simulators with large cross-cockpit display systems, improved data and video networks, and easily reconfigurable cockpits. This paper describes the design concepts and the simulator architecture associated with the CMF and the simulators located in the facility. The Research Flight Deck, the first operational simulator in the CMF is discussed in detail.

Cockpit Motion Facility

Design Concept

Various research programs at NASA Langley Research Center (LaRC) have required motion-based simulation. The Visual Motion Simulator (VMS) has filled that role over the past 26 years by providing a generic cockpit, capable of performing part-task simulations. Over this period research has gradually required more complex tasks and full-mission simulations, with motion, conducted on specific flight decks. Data collected from LaRC programs revealed that about three-quarters of the research tasks could be conducted in a fixed-base mode while the remainder requires motion. Programs usually required that the same cockpit configuration be used for both fixed-base and motion-based studies. The LaRC VMS did not provide this capability. Since many of our research programs did not require full-time motion, it was decided to build a facility that would allow the cockpits to be operated in a fixed-base mode and in a motion-base mode by sharing a common state-of-the-art motion system. This concept fits well with the LaRC philosophy of sharing systems to maximize productivity.

The CMF can support up to four simulators operating simultaneously. Three of these simulators would operate in fixed-base mode while the fourth would operate in either a fixed-base or motion-based mode. Figure 1 depicts an artist's concept of the CMF and shows the locations of these cockpits. The CMF has a central motion-base system that is shared by all of the flight decks. The facility uses the Cockpit Translation System (CTS) to move simulators from their fixed-base site to the motion-base platform. The CTS consists of a 28,000 lb payload crane mounted in the ceiling of the CMF, and a lifting rig to connect the simulators to the crane. This lifting rig is a simple spreader bar with a weight balancing system that connects with the simulators thorough eyebolts located at the bottom four corners of the simulator.

The CMF provides all support systems to the simulators through interface panels that are uniform between the fixed-base sites and the motion-base site. These panels provide hydraulic, electrical, data and video connections. Air conditioning is also uniform between the sites and is provided via an umbilical duct.

---

* *Group Leader, Systems Development Branch, Airborne Systems Competency. Senior Member, AIAA*

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

Motion System

The CMF motion base is a 28,000-pound payload six degree-of-freedom hydraulic motion system. The major components of the system was designed and delivered by McFadden Systems Inc. NASA LaRC is completing the installation and integration of the system including the design of a new digital controller. The payload is limited to 22,000-pounds to increase the performance of the motion-base and simulator. The RFD's weight, without a crew is 20,088 lbs.

Simulators

The CMF facility currently houses three simulators in various stages of completion. The first simulator, the Research Flight Deck (RFD), is an advanced full-mission simulator that is a cross between various types of flight decks. This simulator will be discussed later in the paper. It should be noted that, except for the cockpit instrumentation, the basic structural components and subsystems in the RFD design served as the basis for the remaining simulators. The Integration Flight Deck (IFD) is a "copy" of the flight deck of the NASA Boeing 757 aircraft based at LaRC. Since NASA conducts many simulations in the cockpit of the B-757 it is desirable to have a full-mission simulator based on that configuration to allow research development to proceed in a timely and efficient manner. The IFD has been supporting research programs in a fixed-base capacity and now is being upgraded to motion capable structures and a large field-of-view (FOV) display system. The third simulator is the Generic Flight Deck (GFD). This simulator contains large-size touch panel displays on the instrument panel, the overhead panel and the center console and was designed to provide the researchers with substantial flexibility for addressing cockpit layout issues. This simulator is currently being configured to support research that requires transport-like environments. It is expected to be ready for research in August. The CMF has space for a fourth simulator that is currently under consideration.

The CMF also houses the Research Systems Integration Laboratory (RSIL) contains a copy of the research system on-board the B-757. Each of the simulators can run independently or with the RSIL to provide an integrated simulation environment for conducting complex research programs.

Simulator Design Concept

The design philosophy of the simulation facilities at LaRC is to share common resources in an effort to increase efficiency while reducing costs. The CMF facility, the RFD, and the remaining simulators in the CMF, were designed to this philosophy as much as possible. This approach yields benefits in many areas by maximizing interchangeability, reducing maintenance training and repairs, and providing a common interface to the motion-base support systems. Commonality in all of the CMF simulators includes the overall architecture for all simulators includes structural components, support systems, cockpit system, air conditioning, display systems, and power systems.

Simulator Structures

Each simulator in the CMF has the following major structural components: The mating base, aft-enclosure, cockpit, display system, and floor structures. The loading requirements for all simulator structures are derived from the CMF motion systems capability. The maximum loading requirement for these simulators is:

Vertical sine wave oscillations:
- 0 - 2 Hz up to 1.5g
- 2 - 10 Hz from 1.5g tapering to 0.75g on a logarithmic frequency scale.

Lateral/Longitudinal sine wave oscillations:
- 0 - 2 Hz up to 1.0g
- 2 - 10 Hz from1.0g tapering to 0.75g on a logarithmic frequency scale.

The structures are capable of sustaining a 2.5g shock with a 1.5 amplification factor or 3.75g. The structures were built with a safety factor of 2.0 for material yield and 2.5 for material ultimate strength. All structural components have detailed Finite Element Analysis (FEA) models that are integrated with the CMF motion system to accurately identify loads and fundamental frequencies prior to manufacture. During the manufacturing process several structural components were tested with respect to the FEA model and found have been accurately represented. Modal testing is planned to identify mode shapes and natural frequencies for the integrated structures.

Mating Base: Each simulator is constructed on top of a mating base (figure 2, item 1). The mating base is a steel structure (204" x 200" x 5.25") designed to serve as the interface between the simulator and the CMF motion-base. It provides the substructure for the simulator. All simulator structures are mounted on top of the mating base. Eyebolts are located at the four corners of each mating base and are used by the Cockpit Translation System to move the simulator between fixed-base and motion-base locations.

Aft-Enclosure: The aft-enclosure houses most of the support systems used for simulation (figure 2, item 2). Each simulator has an aft-enclosure that has eight stan-

dard 19" racks, each 72" high, designed to support up to 300 lbs of equipment per rack (figure 3). Therefore the maximum payload in the racks could be as much as 2400 lbs. However, the maximum total electronics rack weight was limited to 1750 lbs to reduce the overall weight of the simulator. This approach allows more freedom in component placement within the racks. The construction of the RFD's aft-enclosure is a steel/aluminum frame with bolted connections to allow for more accurate FEA representation. The steel structure supports the majority of the load imparted by the display system projector units mounted above the aft-enclosure and the aluminum components support the electronics racks. A 5" aluminum false floor is installed inside the aft-enclosure area to route cables between the racks and the cockpit. Three seats are mounted in the aft-enclosure. The seats are C-130 airplane replacement seats that move fore/aft on tracks. These seats are located in front of racks 1 and 8 and just behind the center console in the cockpit.

Cockpit: The cockpit structure in each of the simulators in the CMF is approximately the same. The cockpit is constructed from aluminum box beams and aluminum exterior skins. The shape of the inside of the cockpits is based on the approximate size of a B-757 airplane cockpit. Interior components and interior skin for each simulator provide the desired vehicle dimensions. The exterior skin of each of the cockpits is faceted to ease fabrication (figure 4). The windows are cut to approximate the pilot and copilot FOV from the forward four windows of a B-757 airplane. The FOV that the pilot sees from the eyepoint is shown in the Aitoff plot in figure 5. The copilot's FOV is the mirror of the pilot's FOV. The pilot and copilot's eyepoints are offset by 21" from the centerline and are both 30" forward of the aft of the cockpit. Like the floors in the aft-enclosure, a 5" aluminum false floor is installed in the cockpit to route cables between the cockpit instrumentation to the support systems installed in the aft-enclosure racks. The pilot and copilot seats are mounted on the cockpit floors in the RFD. These are refurbished and reupholstered B-737 airplane seats. The eyepoints are 47.5" above the false floor.

Display System Structures: Each simulator has an out-the-window display system. The RFD and IFD have a Panorama Display System. Several major structural components are part of that display system and include the mirror cell (figure 2, item 3), the projector platform assembly (figure 2, item 4), the Back Projection Screen (hidden inside the mirror cell in Figure 2), the canopy and pedals (figure 2, item 5). The GFD uses four Wide-Angle Collimated (WAC) displays for its out-the-window display system.

Common Simulator Subsystems

Each simulator has generic subsystems to provide basic support to the simulation. These systems include power distribution, hydraulics, air conditioning, voice and data communications, sound, video distribution, smoke/fire detection and an Engineering Workstation to control the simulation. All signals, power, hydraulics and air conditioning are fed to the simulator through the interface panel and are routed in the aft-corners of the aft-enclosure for distribution to the appropriate rack.

Power Distribution: Each cockpit has a single 240 volt (V) input for all electrical power. This input is routed to the power distribution equipment located in rack 4. The power is then converted to the various voltages required by the cockpit including D/C voltages of 28V, 5V and ±15V and A/C voltages of 115V at 400Hz and 120V at 60Hz. The 120V power is distributed via individual power distribution panels located in each rack. Rack 7 contains an additional ±10V DC source to provide reference voltages for the data communications system also located in rack 7.

Hydraulic System: The hydraulic systems associated with each cockpit are routed from the interface panel, through the aft-starboard corner and then to the accumulators and manifolds located under racks 1, 2, 7 and 8. The operating pressure is approximately 1550 psi. Hydraulic power is then routed to each particular control inceptor in the cockpits. The McFadden Digital Control Unit located in rack 3 controls the hydraulic power system.

Air Conditioning System: The air conditioning system maintains positive pressure in the cockpit and the mirror cell to minimize dust accumulation in the display system. The air flows in the aft-port corner of each cockpit and is then routed to the crew and equipment via a duct system containing blast gates to regulate airflow. The air system in each of the simulators is tailored to the equipment and heat load expected in that simulator.

Voice Communication System: The voice communications system in each of the simulators provides five audio stations capable of using standard crew headsets. Each audio station has a control panel that provides simulated access to three VHF and three HF radios in addition to multiple channels of intercom and public address. The system receives volume control and channel/frequency selection information for up to 16 NAV radios (e.g. ILS, DME, VOR, etc). The voice communication system is under simulation host computer control and can be loaded with predetermined

setups for Cockpit Voice Recorders and input radio frequency selections.

Data Communications System: The data communications system used in the CMF consists of VME chassis interconnected with SCRAMNet+. Each simulator has one or two VME chassis's loaded with interface cards for the equipment installed in the cockpit. Typical installations require many types of data to be converted to SCRAMNet+, which is then routed to the simulation host computer and other external devices. The RFD has the following signals (used/available channels):

Discretes In (348/512)
Discretes Out (377/512)
Analogs In (15/32)
Analogs Out (37/64)
ARINC 429 Signals (19/32 Rcv, 20/32 Xmt)
RS232A Signals (4/16)

All of these signals are converted to the SCRAMNet+ format and sent to the host computer through a single fiber optic connection. This greatly simplifies the simulator interface requirements. Ethernet communications are provided for the Engineering Workstation and the VME chassis to support system setup and maintenance tasks.

Sound System: Each simulator contains a MIDI-based (Musical Instrument Digital Interface) sound system to provide cockpit aural cues and advisory and warning messages. Speakers are located in rack 3 and 6 as well and in the cockpit area. The sound system is under host computer control and can drive up to 23 audio channels simultaneously. These sounds are selected from a library of sounds previously loaded on the system. The sounds can be computer generated or actual recordings. The sound system is driven though the VME chassis.

Video Distribution System: Video for the out-the-window display system and cockpit instrumentation is provided to the simulator via a fiber optic video distribution system that routes signals from the remote computer locations to the CMF. Fiber optic transmission is required due to the large distances and the required signal bandwidths. The RFD has ten channels of video for the cockpit instrumentation displays (eight heads-down-displays, one heads-up-display, one spare) and five channels of video for the out-the-window scenery. The total number of conductors used for the 15 channels of video is 75 but the number of connections required at the interface panel is minimized by judicious use of multi-conductor connectors. Once the signals are routed through the interface panel they are converted in racks 6 and 7 to electrical signals. The out-the-window video is sent directly to the projectors located on top of the aft-enclosure. The instrumentation display video is amplified before it is sent to the cockpit displays. All simulators support two internal cameras to record cockpit activities. To allow mixing with out-the-window and instrumentation video, the camera output is sent from the cockpit to the video laboratory.

Smoke and Fire Detection System: Each simulator is equipped with a sophisticated fire detection system. A network of "sniffers" is installed in the racks and cockpit areas to provide early warning of possible trouble. The system is interconnected with the CMF fire system and the LaRC fire station.

Engineering Workstation: The Engineering Workstation (EWS) [1] is located in rack 8. The EWS is a Silicon Graphics Incorporated workstation, hardened for a motion environment, running software that allows it to interface to the host computer. Typically a software engineer will initiate and operate the simulation from this location. This is advantageous because it allows the software engineer to be more interactive with the researchers to control the simulation and more effectively identify and resolve problems.

Research Flight Deck

The RFD is an engineering simulator that is representative of a state-of-the-art advanced subsonic transport airplane (figure 6) [2]. The RFD combines some of the best characteristics found in state-of-the-art transport airplanes including the Airbus series, the Boeing 777, 747-400, MD-11 and in-house research conducted on the NASA 737 Transport Systems Research Vehicle [3]. A B-757 aircraft math model typically drives this advanced flight deck although other models may be substituted. The flight deck systems are designed to be fully reconfigurable. The RFD is used to evaluate and refine research concepts in a high fidelity, full-systems flight operations, two-crew setting from engine startup to engine shutdown. The RFD consists of five total positions: two crew positions, one software engineer position and two observer/researcher positions. Simulating ships systems provides full mission functionality. The architecture of the data communications system and the design of the cockpit systems make aircraft line-replaceable units easy to support. This dramatically enhances the simulators ability to integrate new technology and provide simulation testing on proposed aircraft equipment.

Cockpit Systems

Main Instrument Panel (MIP): The MIP has eight independent ARINC D size displays (numbered in figure 7). These displays are raster CRTs drawing the instru-

mentation at a resolution of 1024x768 pixels at 60 Hz. Although any display format can be drawn on these CRTs the baseline formats on displays 1 and 8 are pictorial representations of aircraft subsystems (electrical, hydraulic, etc.) including status information; on displays 2 and 7 are the Primary Flight Displays (PFD) formats independently configured; on displays 3 and 6 are the Navigation Displays (ND) independently configured; on displays 4 and display 5 are the upper and lower EICAS (Engine Indicating and Crew Alerting System) formats. The MIP also supports the landing gear control, which is a refurbished B-757 landing gear unit. Electromechanical standby instrumentation is located to the right of the pilot's ND display. An autobrake control panel and a pressure meter are also located on the MIP.

Glareshield (GS): The GS, located directly above the MIP, has the same form and overall size as found on the B-757 airplane. The structure under the GS is capable of housing a standard B-757 Mode Control Panels (MCP) (figure 7) or a large research specific MCP. The GS also houses an experimental systems clock that is under host computer control, Master Caution/Warning lights, and audio switch/clock/light panels located on the two outboard wings.

Overhead Panel (OHP): The OHP contains control panels and switches for each of the systems generally found on a modern jet transport airplane (figure 8). Most of the components are refurbished aircraft components that were modified for use in the simulator. The panels installed are:

Windshield Clear	Annunciator
Equip Cooling	Anti-Ice
Hydraulic	Fuel Control
Electrical Systems	Battery Standby
Emergency Lts/Passenger	Window Heat
MIP Panel/Flood Lighting	Bleed Air
Dome/Overhead Lights	Passenger Signs
Engine Start/RAT	Anti-Collision Lights
IRS Mode Select	Exterior Lighting
Cabin Altitude Control	Yaw Damper
Cabin Pressure Indicator	Packs Air
Cockpit Voice Recorder	APU Start
Fuel Quantity Indicator	SELCAL
APU/Cargo Fire Control	Compartment Temp
Engine Fire Control	

These units can be easily moved to meet the configuration requirements of the research program. All panels are under host computer control except for those that control interior cockpit lighting.

Center Control Stand (CCS): The CCS consists of a simulated B-757 airplane throttle quadrant and flight control inceptors (figure 9). Left and right engine thrust controls and the flap controls are independently electrically backdriven to provide autopilot capability. The throttle quadrant is also equipped with reverse thrust, spoiler handles, TOGA buttons, autothrottle disconnect buttons, and trim controllers and indicators. Fuel control levers are also provided to allow full mission simulation from engine startup to shutdown. A parking brake is also provided in the CCS. The CCS also has several electronic panels including:

NAV/PFD Display Cont (2)	MFD Control (2)
Touchpads (2)	VHF COMM (2)
Audio Select Panel (2)	EICAS Control
TCAS/Transponder	Rudder Trim Control
Stabilizer Trim Indicator	Display Select

Several of these are actual aircraft components while many of them are researcher specified.

Heads-Up-Display (HUD): The RFD supports a HUD manufactured by the Flight Dynamics Corporation. The facility has 2 HUD's that are shared among the B-757, the IFD simulator and the RFD simulator for various research programs. A raster to calligraphic conversion box, mounted in rack 7, is required between the HUD and the host computer. A custom set of cockpit interior skins covers the HUD when it is installed.

Control Inceptors: There are a number of control inceptors, in addition to the throttle quadrant, in the RFD including:

Sidearms (2)	Rudders (2)
Tiller	Toe Brakes (2)
Autobrakes (2)	Parking Brake
Aileron Trim Control	Rudder Trim Control
Stabilizer Trim Control	Fuel Control Levers

The RFD sidearms and rudders are hydraulic control loaded systems manufactured by McFadden Systems Incorporated (figure 10). The force-feel characteristics are digitally controlled from a specialized PC located in rack 3, called a Digital Control Unit (DCU). The sidearms and rudders can be operated independently or cross-coupled as required. All parameters for the math model running on the DCU can be downloaded from the host computer to configure the sidearms for the research task. The maximum force available is 50 lbf. Rudder pedals, with toe brakes, are provided for each pilot. The rudder pedals have an electric jackscrew system that allows the pedal position to be moved fore and aft as the pilot and co-pilot requires. Emergency hydraulic and electric shut off switches are on two Emergency Control Panels located within reach of ei-

ther pilot. An electrically control-loaded steering tiller provides nosewheel steering for the RFD (figure 10).

Display system

The display system is composed of the out-the-window scenery computers and the projection system. The CMF uses Evans and Sutherland Image Generators (IG) to provide out-the-window scenery. The RFD can be driven by any of 5 IGs all with approximately the same capabilities including the ESIG 3000GT, ESIG 4530 and Harmony series systems. Some of the salient features of the IG are:

1 M pixels/channel
3500 polygons/field at 60Hz per channel
Extensive use of generic and photospecific texture
1000 calligraphic lights/channel at 60 Hz
Real-fog
Storm, rain and lightning effects
Enhanced visibly effects
Multiple cloud layers
Landing light lobes, steerable lobes
Continuous time-of-day, horizon effects
Proper occultation
Advanced and Fast-Jet collision detection

The research programs can choose between several databases including:

East Coast Database
High Altitude Research Database (Edwards AFB)
Shuttle Mission Landing Sites (15)
Atlanta (ATL)
Boston (BOS)
Chicago (ORD)
Dallas-Ft Worth (DFW)
Denver (DIA)
Los Angles (LAX)
San Francisco (SFO)
Seattle (SEA)

The projection system is a SEOS Panorama display system. The Panorama display system was installed to allow research programs to have cross-cockpit viewing. The system installed in the RFD is a 200° x 40° field-of-view system capable of projecting raster displays with calligraphic lights at 2 arc-min/pixel resolution. The RFD's Panorama is rotated down 3° about the eye-point resulting in a vertical field-of-view of +17°/-23°. This allows the researcher to have more visual scene over the nose of the aircraft to support several research programs that require extensive amounts of ground taxi tasks and approach to landings.

Simulation Model

The math model typically used in the RFD is representative of a B-757 airplane. Other vehicle models may be used. The software controls the RFD hardware through the data communication system and the VME chassis. Just as the simulator hardware is modular and can easily accommodate new components, the object-oriented design of the software allows the baseline math model to easily interface new components [4]. The use of common subsystems across simulators has enhanced software reusability and allowed the software development staff to realize a gain in productivity.

Summary

The shared resource approach provided by the CMF will allow NASA LaRC to effectively conduct fixed-base and motion-base research on a number of specific cockpits simultaneously. The common subsystem design significantly increases the maintainability of the simulators and allows software developers to easily transition from one simulator to another. The RFD is a state-of-the-art engineering/research simulator combining some of the best characteristics of several different aircraft platforms and NASA research programs. The RFD has completed several studies to date and is currently supporting development for five research programs. The RFD is ready to support program requirements now and has the flexibility to support future program requirements.

References

[1] Kaplan, J.A. and Kenny, P.S., "SIMGRAPH – A Flight Simulation Data Visualization Workstation". AIAA-97-3797. Modeling and Simulation Technologies Conference, New Orleans, LA, August 11-13, 1997

[2] Knox, C.E, "The NASA Transport Research Facilities Requirements Document, v3.4". NASA Langley Research Center, Hampton, VA, February 1997

[3] Wallace, L.E., *Airborne Trailblazer, Two Decades With NASA Langley's 737 Flying Laboratory*, The NASA History Series, NASA SP-4216, 1994

[4] Leslie, R.A., Geyer, D.W, Cunningham, K., Glaab, P.C., Kenney, P.S., Madden M.M., "LaSRS++ An Object Oriented Framework For Real-Time Simulation Of Aircraft". AIAA-98-4529. Modeling and Simulation Technologies Conference, Boston, MA, August 10-12, 1998

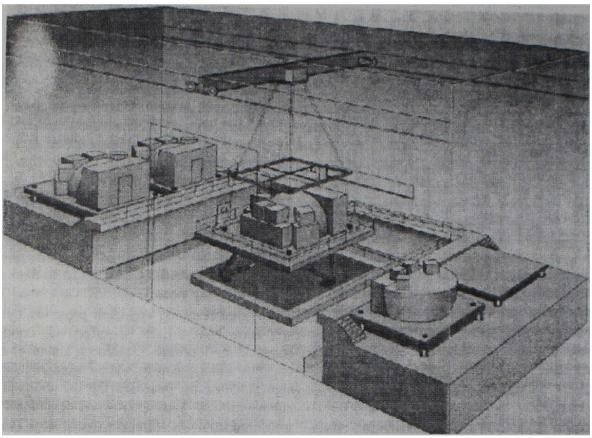


Figure 1 – Cockpit Motion Facility

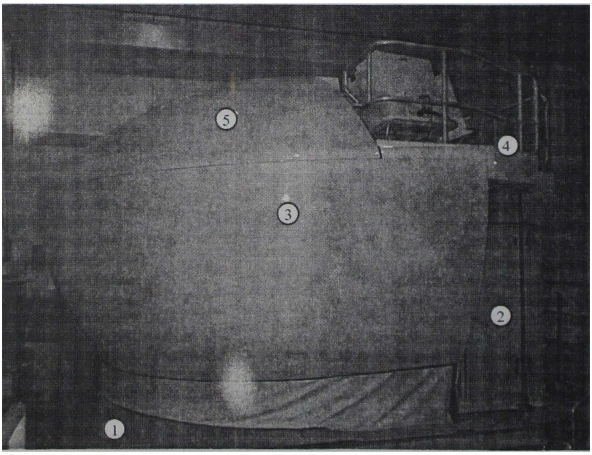


Figure 2 – RFD Exterior

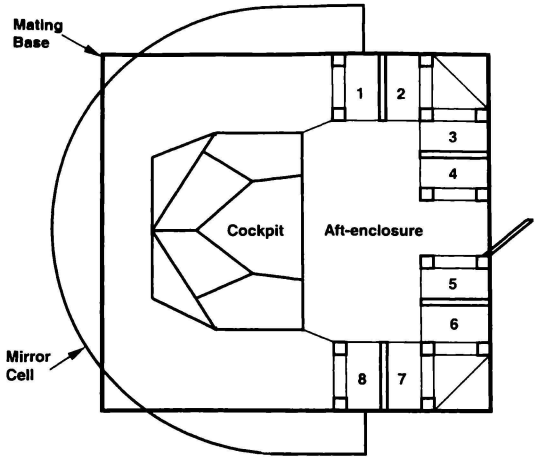


Figure 3 – Simulator Plan View

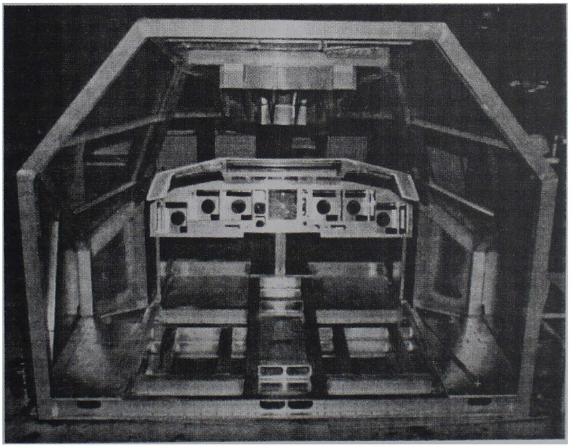


Figure 4 – RFD Cockpit Shell

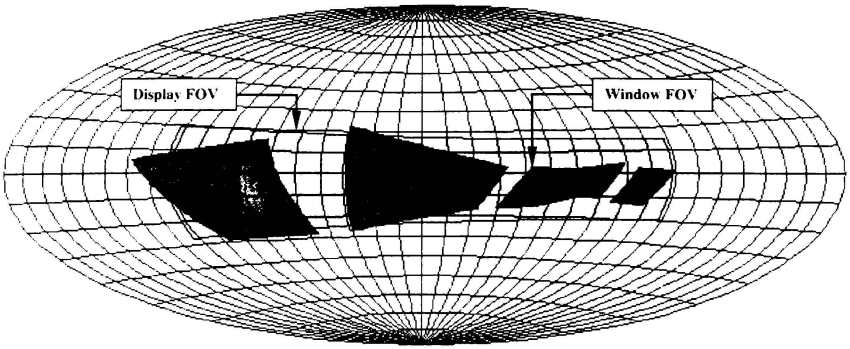


Figure 5 – Aitoff Plot (Pilot's Eyepoint)

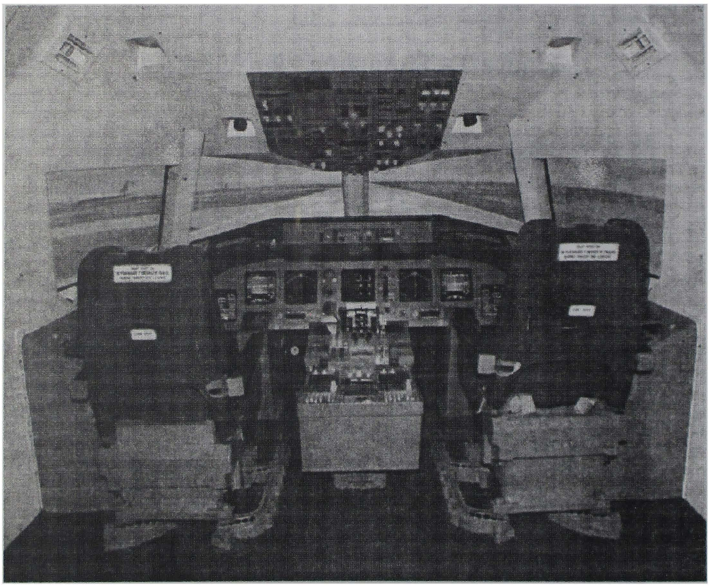


Figure 6 – RFD Interior

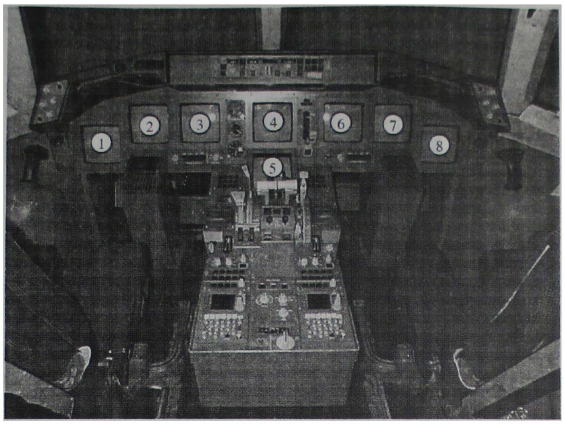


Figure 7 – RFD MIP/GS

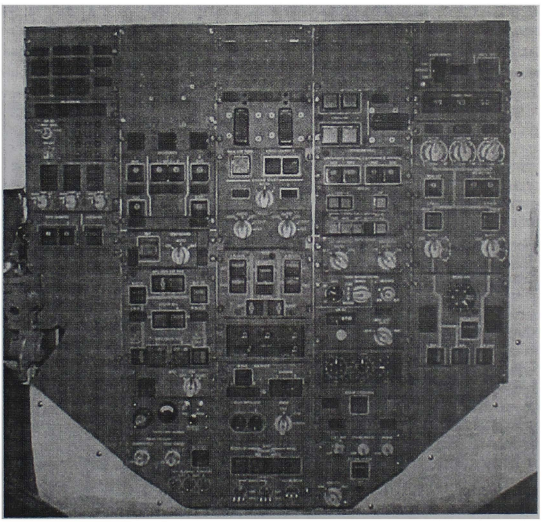


Figure 8 – RFD OHP

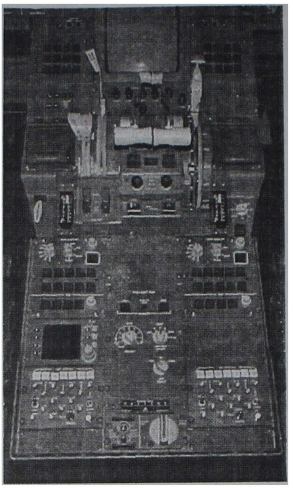


Figure 9 – RFD CCS

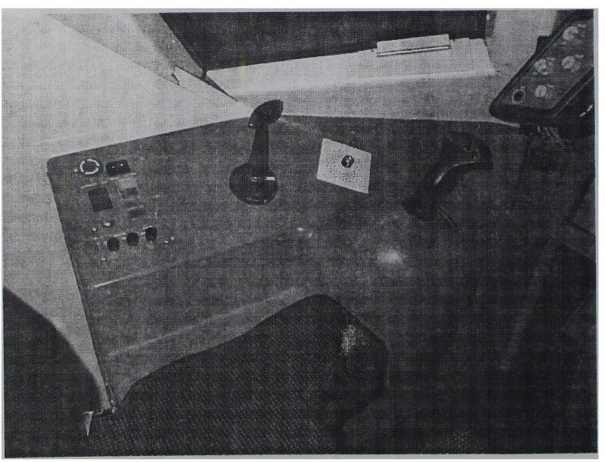


Figure 10 – RFD Side Console

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37235

DESDEMONA: ADVANCED DISORIENTATION TRAINER AND (SUSTAINED-G) FLIGHT SIMULATOR

AIAA-2000-4176

Willem Bles, Ruud J.A.W. Hosman and Bernd de Graaf
TNO Human Factors
P.O.Box 23
3769 ZG Soesterberg, The Netherlands
bles@tm.tno.nl

Abstract

Desdemona is a sophisticated demonstration, simulation and training facility specified by TNO Human Factors and developed by AMST Systemtechnik. The Desdemona concept adds a sustainable G-load to the conventional Stewart platform motion, without the angular accelerations encountered in centrifuges during the G-onset. This is accomplished by having the aviator sitting in a fully gimbaled cockpit with four cascaded degrees of freedom (360 degrees of yaw, pitch and roll rotation, and heave (2m)) on a placed on a longitudinal track (8m) which is rotated around the vertical axis adding two synergistic degrees of freedom. Rotation of the track allows centrifugation up to 3g. Realistic environment with state-of-the-art visuals and aircraft specific instruments is available in the cockpit. Proper combination of the 6 DOF makes Desdemona a dynamic flight simulator.

Introduction

For reduction of costs, for environmental reasons and for safety reasons, simulators will be used more and more in the near future. Dependent on the training goal, simulators can be relatively simple and small for procedure training, or complex and large in case the simulation requires a motion base. The most common motion platform is the Stewart six-pod platform, which allows all six degrees of freedom. For many applications such as training in transport aircraft this platform is sufficient, despite the fact that accelerations can be induced only for a short time. However, for fighter aircraft that are also pulling sustained high G-loads, this platform is not sufficient for realistic mission rehearsal. Present developments show that for flight simulation with higher G-loads, centrifuges are built with roll and pitch options of the gondola. Problems with these simulators are that they are able to simulate the G-load, but that the corresponding angular accelerations during the onset introduce conflicting motion information to the pilots equilibrium system, easily leading to unwanted side effects like disorientation and motion sickness. The proposed Desdemona-concept is a research simulator, which combines the possibilities of the common Stewart platform with the possibility of the sustained G-load, however, without the co-varying rotational accelerations. In this paper the advantages of the Desdemona-concept as a research tool are further elaborated as well as its commercial capabilities.

Current developments in military aircraft: superagility

Modern technologies like thrust vectoring control, canards, fly-by-wire techniques, allow the pilot to maneuver in the post-stall regime. The consequence is that the pilot will no longer be exposed to Gz-forces only, but Gx and Gy will be encountered as well. Indications of the order of magnitude for Gx and Gy are about 6.5 en 4 g respectively. Since at the same time there is an increase in the rotation capabilities of the aircraft, adequate handling of the motion information by the pilot's equilibrium system is at risk. A consequence of post-stall maneuvering is that the attitude of the aircraft and the velocity vector do not necessarily correspond to each other. Common flight instruments are therefore insufficient to help the pilot to keep a proper spatial orientation. Flying with closed-cockpit to avoid laser threats and off-boresight targeting options even more increase the chance on spatial disorientation and motion sickness. Advisory committees on human factor aspects of these developments propose therefore *ground-based training* and *demonstration* of the sensory consequences of these adverse motion environments[1].

"Copyright © 2000 by TNO Human Factors. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

It is especially in this respect that the Desdemona concept may prove its value.

The Desdemona concept

The Desdemona concept was developed by TNO Human Factors (The Netherlands) in conjunction with AMST Systemtechnik (Austria). In Fig. 1 the concept is shown. It consists of a fully gimbaled cockpit, which allows for unlimited rotation in all directions. The pilot, with his head in the center of rotation, controls the flight instruments and has a view outside via up to date visuals. For simulation purposes different aircraft models and databases are available. The cockpit can move along an 8 m horizontal track. The cockpit can also move vertically over 2 m. Both horizontal and vertical accelerations allow for 0.5g. Finally, the horizontal track is rotated around a vertical axis, which means that centrifuging is also possible. The working principle of a normal centrifuge is variation of the angular velocity with constant eccentricity, resulting in a varying G-load. However, keeping the angular velocity constant and varying the amount of eccentricity, is another way of varying the G-load. Desdemona applies both with a maximal G-load of 3g.

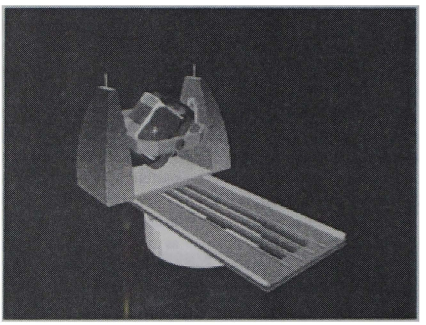


Fig. 1 Desdemona

The main characteristic of the concept, i.e. the variation of the G-load by varying the eccentricity, has consequences for simulation. It means that the onset of a sustained G-load is not necessarily accompanied with a strong angular acceleration, as is the case in the conventional centrifuge. It is possible to start the rotator subliminal up to the desired speed with the cockpit in the center position, and subsequently move the cockpit away from the axis without varying the angular velocity. It is obvious that one should take into account the linear Coriolis accelerations during the movement of the cockpit on the track.

The second characteristic is that the cockpit can be moved in the vertical direction as well. This offers exceptional good possibilities to vary Gx or Gy during a sustained Gz load almost without concurrent variation of the angular motion of the cockpit. This is especially of importance for the demonstration and training aspects of motion perception during post-stall maneuvering.

Application of the Desdemona-concept

Desdemona as disorientation demonstrator

As mentioned above, spatial disorientation is easily induced in superagile aircraft. It is therefore of paramount importance that pilots will encounter these motion illusions in a disorientation demonstration refresher course. In demonstrating these motion illusions on the ground-based device the same mechanisms of the human equilibrium system have to be involved that are responsible for the origination of the illusion during the real flight situation. This requires sufficient knowledge of the human equilibrium system[2,3]. The Desdemona concept is the ideal motion system for this sort of

demonstrations. This is not a coincidence because that was the main drive for the design specifications. It should be noted that there are presently no other demonstrators available in the market that are able to demonstrate explicitly the illusions common to post-stall maneuvering. The commercially available small demonstrators can barely simulate vestibular illusions, the more advanced demonstrators have fully gimbaled gondolas, but neither a varying arm length for eccentric rotation, nor a vertical motion component.

Desdemona as simulator

The Desdemona concept allows for different simulator modes. The simplest one is the fixed base simulator mode. The second mode is a sort of Stewart platform motion: With the cockpit in a halfway eccentric position, the simulator can be programmed to move as any other motion platform, Fig. 2. With the cockpit in eccentric position Desdemona is in the centrifuge mode, Fig. 3. During flight simulation transfer from one mode to another will be possible. It is recognized that this can only be realized within some boundaries (the envelope of which will be explored). For realistic post-stall maneuvering, where the G-load is not very high, maneuvers can be flown with a high level of realism with respect to the G-load and the stimulation of the human equilibrium system. Due to the experience of TNO with testing subjects with varying eccentricity on the Coriolis Acceleration Platform at Pensacola, Florida, we believe that the Desdemona concept allows for realistic flight simulation[4,5].

Desdemona as test-bed

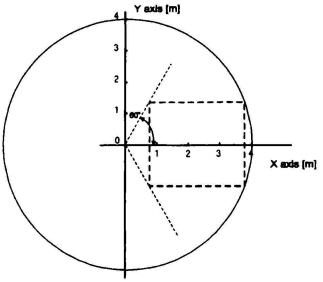
For applications of 3D audio, new speech technologies, novel interfaces or Smart Suits (autorecovery system), the Desdemona concept is an ideal test-bed. This also applies for head mounted displays (closed cockpit concept), not only in terms of their effect on task performance, but also in terms of ergonomic design and medical implications. This is the more true since forces are not only directed along the subject's z-axis, but also along the intra-aural axis, exerting a heavy load on the pilot's neck.

Miscellaneous applications

- Aeromedical research and training can be done elegantly with the Desdemona concept.
- Driving simulators lack the facility to generate sustained G, which is apparent in the problems experienced in driving simulated curves. Desdemona could be useful to sort out the necessary information on the minimal G-load required to reach transfer of training and, at the same time, how to minimize simulator sickness.
- Low cost simulation aims at simple simulators, maintaining transfer of training. With respect to motion it needs to be established what really is necessary on motion information for the various training goals. This holds for driving simulators as well as for flight simulators. It is obvious that the Desdemona concept with the different simulator modes can be used to sort this out.
- The Desdemona concept is also a simulator with promising options to serve as a helicopter simulator because of the vertical stroke and the unlimited rotatory motion possibilities.

References

1. Welsch, H., Albery, W., Banks, R.D., Bles, W. (2000). Physiological consequences: cardiopulmonary, vestibular, and sensory aspects. *RTO HFM Lecture Series on "Human Consequences of Agile Aircraft"*, RTO-EN-12.
2. Bles, W. (1998) Corolis Effects and Motion Sickness Modelling. *Brain Res. Bull.*, 47 (5): 543-549.
3. Groen, E.L., Hosman, R.J.A.W., Clari, M.V. (2000) Psychophysical thresholds associated with simulation of linear acceleration. AIAA conference, Denver August 2000, USA
4. Graaf, B. de , Bos, J.E., Tielemans, W., Rameckers, F., Rupert, A.H., Guedry, F.E. (1996) Otolith contribution to ocular torsion and spatial orientation during acceleration. *Naval Aerospace Medical Research Laboratory.* NAMRL Technical Memorandum 96-3, Pensacola Fl, U.S.A.
5. Graaf, B. de, Bos, J.E., Groen, E., Tielemans, W., Rameckers, F., Clark, J.B., Mead, A.M., Guedry, F.E. (1998) Otolith responses during centrifugation along three axes of orientation. *Naval Aerospace Medical Research Laboratory,* NAMRL-1402, Pensacola Fl., U.S.A.



Desdemona 'Stewart' Mode	Amplitude [m]	Velocity [m/s]	Acceleration [m/s^2]
X, surge	*0.78/ +*	*± 1.50*	*± 2.45*
Y, sway	*± 1.*	*± 0.41*	*± 0.82*

Fig. 2 Desdemona in the 'Stewart' Mode

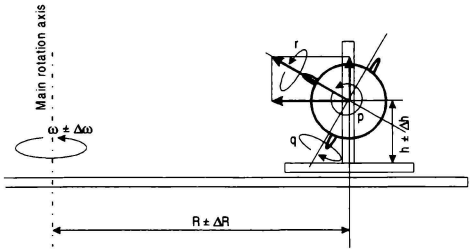


Fig. 3 Degrees of freedom in the Centrifuge Mode

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37236

AIAA 2000-4177

SIMULATION OF THE DYNAMICS OF SPACE SYSTEMS USING OBJECT-ORIENTED MODELING

Byung No Min * and Arun K. Misra †
McGill University, Montreal, Quebec, H3A 2K6, Canada.
Xavier Cyril ‡
CAE Electronics Ltd., Saint-Laurent, Quebec, H4L 4X4, Canada.
Vinod J. Modi § and Clarence W. de Silva §
University of British Columbia, Vancouver, British Columbia, V6T 1Z4, Canada.

This paper presents a practical formalism for modeling the dynamics of rigid/flexible multibody systems having either a non-fixed base or a fixed base. The mechanical components of the multibody systems are represented by modular objects with appropriate characteristic variables and mathematical operations. In order to conveniently accommodate flexible bodies as well as maintain the modular representation for individual bodies, the formalism is based on a variation of the Lagrange method that utilizes the natural orthogoal complement of the kinematic constraint matrix. The objects are implemented in an object-oriented software environment and an actual modeling is realized by the interconnection of these objects in a simple and clear manner. Modification in the architecture of a system can be accomplished quite easily in this modeling process, and various types of configurations can be simulated, including flexible tree-type structures in the free-floating mode. Several simulations are performed, through which the proposed modeling approach is validated.

Introduction

Space robotic manipulators and satellite systems belong to the regime of flexible multibody systems whose dynamics is well known to be complex and challenging. Considerable effort has been directed towards dynamics modeling and computer simulation of these systems because the micro-gravity environment of space is not easily amenable for experimentation on the ground. Most of the previous investigations, however, have considered specific models and have followed the procedure-oriented approach, in which the modeling is composed of a large number of procedures. Recently, several researchers have paid their attention to the application of object-oriented concepts[1] to multibody dynamics formulations because it is believed to have a lower probability of errors in modeling and makes it easy to upgrade or modify already developed systems by simply adding or replacing some objects.

Otter et al.[2] introduced the object-oriented concept into a data model for the exchange of rigid multibody system descriptions. They classified multibody elements into part and interaction classes and organized them in a hierarchical framework. Wallrapp[3,4] extended the work of Otter et al. and appended data classes for flexible members on the basis of the modal representation. Both of these works, however, aimed at preparing a standard input data form for various multibody software packages and did not consider a method to formulate the equations of motion.

Lückel et al.[5] adopted a recursive Newton-Euler algorithm and grouped the corresponding functions into hierarchical modules. They expected a possibility of integrating the function modules into an object-oriented modeling system. Kecskeméthy[6] was mainly concerned with the application of object-oriented computational techniques to the efficient generation and solution of the equations of motion for **rigid** multibody system. He adopted the idea of using the inverse dynamics repeatedly for the derivation of the direct dynamic equations. He regarded a multibody system as a composition of two categories of objects: state objects that holds information about the motion and loads, and transmission elements that transmit the information from one state object to another. His work, however, was confined to rigid body systems.

Anantharaman[7] classified the elements of multibody systems into kinematic, dynamic, and constraint types. The work-energy representation was used for dynamics. Although he mentioned the possibility of incorporating the stress measure of a flexible member

*Graduate student, Dept. Mechanical Engineering
†Professor, Dept. Mechanical Engineering, AIAA memeber.
‡Manager, Dept. Space System Engineering
§Professor, Dept. Mechanical Engineering, AIAA member.

Copyright © 2000 by B. Min, A.K. Misra, X. Cyril, V.J. Modi, and C.W. de Silva. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

through the virtual work, he did not develop a rigorous formulation.

The literature survey shows that most of the research has been confined to rigid body systems having a fixed base and it is hardly suitable for the space applications.

The primary objective of this paper is to present a simulation tool for **flexible** tree-type multibody systems with or without a fixed base by adopting the object-oriented concept. The goal is to identify characteristic attributes of the elements of flexible multibody systems and to realize them into modular objects having certain characteristic parameters and operations. In order to conveniently accommodate flexible bodies as well as maintain the modular representation for individual bodies in a system, the formalism is based on a variation of the Lagrange method that utilizes the natural orthogonal complement of the kinematic constraint matrix.[8]

Consequently, the paper provides a theoretical base for modeling flexible members in multibody systems in a modular fashion. Also, a practical formalism is established for the implementation of the mechanical components on an object-oriented programming environment. Modeling is achieved by the interconnection of the object modules in a simple and clear manner. Modification in architecture of the system can be accomplished quite easily in this modeling process, and various types of configurations can be simulated, including flexible tree-type structures in the free-floating mode. Several simulation examples are presented in the paper.

Theoretical Considerations

A multibody system is composed of various types of bodies and joints, which have distinctive characteristics and functionality. Dynamics modeling of such a system can be treated as interconnection of the corresponding functional modules. Each of modules is eventually cast into a small independent unit called an "object". The development of the multibody objects is based on two generic principles, those of kinematics and dynamics. The objects are refined from the generic types in a top-down fashion by taking advantage of the inheritance concept. The theoretical consideration in the object-oriented formalism for flexible multibody systems is discussed below.

Generic elements

Kinematics is a key concept for the description of motion. and kinematic objects are mainly responsible for holding and/or passing the motion states of the components of a multibody system. The fundamental motion states can be position($\mathbf{p}$), velocity($\dot{\mathbf{p}}$), and acceleration($\ddot{\mathbf{p}}_{|\dot{q}=0}$) for translation, and rotation matrix($\mathbf{R}$), angular velocity($\boldsymbol{\omega}$), and angular acceleration($\dot{\boldsymbol{\omega}}_{|\dot{q}=0}$) for rotation. They are held in an object titled *frame*, which in general acts as a reference in expressing various quantities associated with a body. In the above, $(\cdot)_{|\dot{q}=0}$ stands for the evaluation of a quantity when the generalized acceleration of the multibody system are set to zero, and its use makes the assembly of the global system dynamics convenient. It should be mentioned here that an italicized word represents title of an object in this paper.

In accordance with the definition of the object *frame*, the elementary parameters of a generic kinematic object are relative displacement($\bar{\mathbf{r}}$), velocity($\mathring{\bar{\mathbf{r}}}$), and acceleration($\mathring{\mathring{\bar{\mathbf{r}}}}$) for translation, and relative rotation matrix($\bar{\mathbf{R}}$), angular velocity($\bar{\boldsymbol{\omega}}$), and angular acceleration($\mathring{\bar{\boldsymbol{\omega}}}_{|\dot{q}=0}$) for rotation. Here, '$\circ$' over a variable represents differentiation with respect to time within the reference frame of the generic kinematic object.

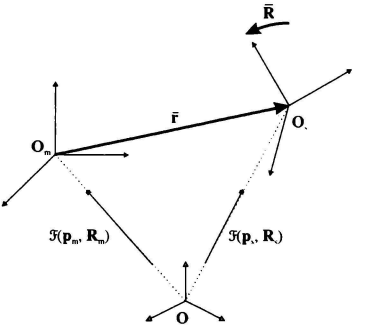


Fig. 1 Generic kinematic element

The fundamental operation of the generic kinematic element is transmission of a *frame* into another as depicted in Figure 1. It constitutes of the following three transmissions.

- displacement transmission

$$\bar{\mathbf{R}}^{\mathrm{T}}(\mathbf{p}_m + \bar{\mathbf{r}}) = \mathbf{p}_s \tag{1}$$

$$\mathbf{R}_m\bar{\mathbf{R}} = \mathbf{R}_s \tag{2}$$

- velocity transmission

$$\bar{\mathbf{R}}^{\mathrm{T}}\left(\dot{\mathbf{p}}_m + \boldsymbol{\omega}_m^{\times}\bar{\mathbf{r}} + \mathring{\bar{\mathbf{r}}}\right) = \dot{\mathbf{p}}_s \tag{3}$$

$$\bar{\mathbf{R}}^{\mathrm{T}}(\boldsymbol{\omega}_m + \bar{\boldsymbol{\omega}}) = \boldsymbol{\omega}_s \tag{4}$$

- acceleration transmission

$$\bar{\mathbf{R}}^{\mathrm{T}}\left(\ddot{\mathbf{p}}_{m|\dot{q}=0} + \dot{\boldsymbol{\omega}}_{m|\dot{q}=0}^{\times}\bar{\mathbf{r}} + \boldsymbol{\omega}_m^{\times}\boldsymbol{\omega}_m^{\times}\bar{\mathbf{r}} + 2\boldsymbol{\omega}_m^{\times}\mathring{\bar{\mathbf{r}}} + \mathring{\mathring{\bar{\mathbf{r}}}}_{|\dot{q}=0}\right) = \ddot{\mathbf{p}}_{s|\dot{q}=0} \tag{5}$$

$$\bar{\mathbf{R}}^{\mathrm{T}}\left(\dot{\boldsymbol{\omega}}_{m|\dot{q}=0} + \boldsymbol{\omega}_m^{\times}\bar{\boldsymbol{\omega}} + \mathring{\bar{\boldsymbol{\omega}}}_{|\dot{q}=0}\right) = \dot{\boldsymbol{\omega}}_{s|\dot{q}=0} \tag{6}$$

In the above, a subscript letter distinguishes a *frame* and the superscript '×' means the cross product operand. It is worth mentioning that $\mathring{\bar{\mathbf{r}}}$, $\mathring{\mathring{\bar{\mathbf{r}}}}_{|\dot{q}=0}$, $\bar{\omega}$, and $\mathring{\bar{\omega}}_{|\dot{q}=0}$ reflect the characteristic features of a flexible object while they vanish for a rigid object. The transmissions mentioned above can be interpreted as a kinematic constraint between the two *frames* and Eqs. (3 - 4) will be used to derive the kinematic constraint equation for specific objects.

Complementary to kinematic objects for multibody modeling, dynamic objects are responsible for yielding the equation of motion of a body, which is usually expressed in the following form:

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbf{w}(\mathbf{q}, \mathbf{v}, t) \tag{7}$$

wherein $\mathbf{M}$ and $\mathbf{w}$ are respectively the inertia matrix and wrench of the body. In most cases they are functions of the generalized coordinate vector $\mathbf{q}$ and the extended velocity vector $\mathbf{v}$, which are provided by the kinematic objects. It should be noted that the extended velocity may not be the direct time derivative of the generalized coordinates but a linear transformation may be involved between the two. As a consequence, the generic dynamic objects are *inertia* representing motion resistance and *wrench* representing motion agent.

Body

Body is an abstraction for parts having mass as well as finite dimension in multibody systems. It may represent a base satellite, a truss arm, a long boom, and so on. It is modeled as a container class that is essentially a combination of objects titled *frame*, *link*, *modal*, *inertia*, and *wrench*.

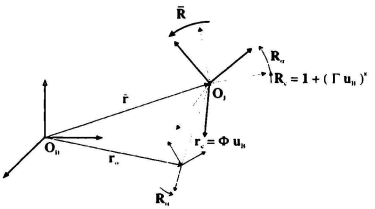


Fig. 2 Generic flexible link

Link is a kinematic object that transfers motion states **within** a body. As depicted in Figure 2, it is characterized by the nominal length vector($\mathbf{r}_o$) and rotation matrix($\mathbf{R}_o$). For a realistic modeling, it should be able to take a deformation due to material flexibility into account, and it can conveniently be achieved by adopting modal representation with the assumption of small elastic deformation. Then, the state quantities of the *link* become:

translation	roation
$\bar{\mathbf{r}} = \mathbf{r}_o + \Phi\,\mathbf{u}_B$	$\bar{\mathbf{R}} = (\mathbf{1} + \{\Gamma\,\mathbf{u}_B\}^\times)\,\mathbf{R}_o$
$\mathring{\bar{\mathbf{r}}} = \Phi\,\dot{\mathbf{u}}_B$	$\omega = \Gamma\,\dot{\mathbf{u}}_B$
$\mathring{\mathring{\bar{\mathbf{r}}}}_{|\dot{q}=0} = \Phi\,\ddot{\mathbf{u}}_{B|\dot{q}=0}$	$\mathring{\bar{\omega}}_{|\dot{q}=0} = \Gamma\,\ddot{\mathbf{u}}_{B|\dot{q}=0}$

wherein Φ is the mode shape matrix of the link and $\Gamma = \left(\frac{\mathbf{r}_o}{\|\mathbf{r}_o\|^2}\right)^\times \Phi$ is the rotational counterpart. $\mathbf{u}_B$ is the modal coordinate vector of the *link* and is provided along with its time derivative by the *modal* object of the body to which the *link* object belongs.

The *link* inherits the kinematic operation from the generic kinematic object so that it transmits a local *frame* from one end to the other. A body reference frame can be located inside the body, in which case one *link* transmits an outside *frame* to the reference *frame* inside and another *link* does vice versa. Hence, *link* has two varieties, inward and outward operation types. They are useful for the resolution of kinematic and dynamic parameters of a body such as mode shapes, moments of inertia, stiffness matrix, and so on, because they offer a freedom in the choice of location of the body reference frame.

The kinematic constraint equation due to the *link* can be obtained by the modification of the velocity transmission relations, Eqs. (3 - 4). For the outward operation type, it becomes

$$\mathbf{X}_L^o \begin{Bmatrix} \dot{\mathbf{p}}_B \\ \boldsymbol{\omega}_B \end{Bmatrix} + \mathbf{J}_L^o \dot{\mathbf{u}}_B = \begin{Bmatrix} \dot{\mathbf{p}}_J \\ \boldsymbol{\omega}_J \end{Bmatrix} \tag{8}$$

where

$$\mathbf{X}_L^o = \begin{bmatrix} \bar{\mathbf{R}}^T & -\bar{\mathbf{R}}^T\bar{\mathbf{r}}^\times \\ \mathbf{0} & \bar{\mathbf{R}}^T \end{bmatrix} \quad \text{and} \quad \mathbf{J}_L^o = \begin{bmatrix} \bar{\mathbf{R}}^T\Phi \\ \bar{\mathbf{R}}^T\Gamma \end{bmatrix}$$

and for the inward operation type, it becomes:

$$\begin{Bmatrix} \dot{\mathbf{p}}_B \\ \boldsymbol{\omega}_B \end{Bmatrix} = \mathbf{X}_L' \begin{Bmatrix} \dot{\mathbf{p}}_J \\ \boldsymbol{\omega}_J \end{Bmatrix} + \mathbf{J}_L' \dot{\mathbf{u}}_B \tag{9}$$

where

$$\mathbf{X}_L' = \begin{bmatrix} \bar{\mathbf{R}} & \bar{\mathbf{r}}^\times\bar{\mathbf{R}} \\ \mathbf{0} & \bar{\mathbf{R}} \end{bmatrix} \quad \text{and} \quad \mathbf{J}_L' = \begin{bmatrix} -\bar{\mathbf{r}}^\times\Gamma - \Phi \\ -\Gamma \end{bmatrix}$$

In the above, $\mathbf{X}_L^o$ and $\mathbf{J}_L^o$ are respectively the spatial transformation and local Jacobian matrices of the outward type *link* and $\mathbf{X}_L'$ and $\mathbf{J}_L'$ are those of the inward type *link*. They play a crucial role in the assemblage of the global kinematic constraint of a whole system. (This will be explained in detail later.) Hence, they are appended to the representative quantities of the *link*.

The dynamic equation of motion of a body can be obtained by the Lagrange method, which conveniently accommodates structural flexibility of the body

through the work-energy principle. The generalized coordinates($\mathbf{q}_B$) of the body are chosen to be the position vector and quaternions of a body reference frame and the modal coordinates. In practice, the equation of motion is expressed in terms of the extended velocity($\mathbf{v}_B$), which are a set of the linear and angular velocity components of the body reference frame and the time rate of the modal coordinates. Since the angular velocity is not the time derivative of the quaternions, the local velocity transformation

$$\dot{\mathbf{q}}_B = \mathbf{\Lambda}_B \mathbf{v}_B \tag{10}$$

is introduced, then the dynamic equation of motion of the body can be written as

$$\mathbf{M}_B \dot{\mathbf{v}}_B = \mathbf{w}_B^S + \mathbf{w}_B^E + \mathbf{w}_B^K \tag{11}$$

where $\mathbf{M}_B$ is the extended mass matrix of the body, while $\mathbf{w}_B^S$, $\mathbf{w}_B^E$ and $\mathbf{w}_B^K$ are wrenches due to the non-inertial and relative motions of the body, the external forces and moments, and the kinematic constraints imposed by the links and joints connected to the body, respectively. The detailed procedure for the derivation of the said dynamic equation of motion can be found in Cyril et al.[8]

Inertia is a dynamic object that constructs the mass matrix $\mathbf{M}_B$ and its time derivative $\dot{\mathbf{M}}_B$. The formulation of $\mathbf{M}_B$ is derived from the kinetic energy expression of the body as follows:

$$\mathbf{M}_B = \begin{bmatrix} \mathbf{M}^{dd} & \mathbf{M}^{dr} & \mathbf{M}^{de} \\ & \mathbf{M}^{rr} & \mathbf{M}^{re} \\ \text{symm.} & & \mathbf{M}^{ee} \end{bmatrix} \tag{12}$$

where

$$\begin{aligned} \mathbf{M}^{dd} &= m_B \mathbf{1} \\ \mathbf{M}^{dr} &= -m_B \mathbf{c}_o^{\times} - (\mathbf{M}^{de}\mathbf{u}_B)^{\times} \\ \mathbf{M}^{rr} &= \mathbf{M}_o^{rr} + [\mu] \\ \mathbf{M}^{re} &= \begin{bmatrix} (\mathbf{c}_{23}^1 - \mathbf{c}_{32}^1) + \mathbf{u}_B^{\mathsf{T}}(\mathbf{C}_{23}^2 - \mathbf{C}_{32}^2) \\ (\mathbf{c}_{31}^1 - \mathbf{c}_{13}^1) + \mathbf{u}_B^{\mathsf{T}}(\mathbf{C}_{31}^2 - \mathbf{C}_{13}^2) \\ (\mathbf{c}_{12}^1 - \mathbf{c}_{21}^1) + \mathbf{u}_B^{\mathsf{T}}(\mathbf{C}_{12}^2 - \mathbf{C}_{21}^2) \end{bmatrix} \\ \mathbf{M}^{ee} &= \mathbf{C}_{11}^2 + \mathbf{C}_{22}^2 + \mathbf{C}_{33}^2 \end{aligned}$$

In the above, $[\mu]$ is a 3×3 matrix whose elements become

$$\begin{aligned} \mu_{ii} &= 2(\mathbf{c}_{jj}^1 + \mathbf{c}_{kk}^1)\mathbf{u}_B + \mathbf{u}_B^{\mathsf{T}}(\mathbf{C}_{jj}^2 + \mathbf{C}_{kk}^2)\mathbf{u}_B \\ \mu_{ij} &= -(\mathbf{c}_{ij}^1 + \mathbf{c}_{ji}^1)\mathbf{u}_B - \mathbf{u}_B^{\mathsf{T}}\mathbf{C}_{ij}^2\mathbf{u}_B \end{aligned}$$

in which i, j, and k are cyclic indices varying from 1 to 3. In the expressions of the mass matrix, the characteristic parameters of the *inertia* can be identified. The total mass(m_B), position vector of the center of mass($\mathbf{c}_o$), and the second moment of inertia($\mathbf{M}_o^{rr}$) of the body are originated from the nominal configuration of the body while the following three

$$\begin{aligned} \mathbf{M}^{de} &= \int_{\mathcal{B}} \mathbf{\Phi} \, dm \\ \mathbf{c}_{kl}^1 &= \int_{\mathcal{B}} r_{o(k)} \mathbf{\Phi}_{(l)} \, dm \\ \mathbf{C}_{kl}^2 &= \int_{\mathcal{B}} \mathbf{\Phi}_{(k)}^{\mathsf{T}} \mathbf{\Phi}_{(l)} \, dm \end{aligned}$$

are due to the structural flexibility of the body. In the above, $r_{o(k)}$ is the k-th component of a nominal position vector $\mathbf{r}_o$ and $\mathbf{\Phi}_{(k)}$ is the k-th row vector of the mode shape matrix $\mathbf{\Phi}$. It should be noticed that the characteristic parameters are constant and evaluated off-line while the mass matrix and its derivatives are functions of the kinematic states of the body such as the modal coordinates $\mathbf{u}_B$.

It is a dynamic object *wrench* that combines all the motion agents on the body and evaluates the right hand side of Eq. (11). As mentioned earlier, each term originates from a different source and contributes to the global dynamics of the system in a different way. The first term is deduced from the derivation of the equation of motion and it is expressed in terms of the kinematic states and inertia of the body as

$$\begin{aligned} \mathbf{w}_B^S = &-\dot{\mathbf{M}}_B\mathbf{v}_B - 2\mathbf{\Lambda}_B^{\mathsf{T}}\dot{\mathbf{L}}_B^{\mathsf{T}}\mathbf{M}_B\mathbf{v}_B \\ &+ \frac{1}{2}\mathbf{\Lambda}_B^{\mathsf{T}}(\mathbf{v}_B^{\mathsf{T}}\frac{\partial \mathbf{M}_B}{\partial \mathbf{q}_B}\mathbf{v}_B) - \mathbf{\Lambda}_B^{\mathsf{T}}\frac{\partial U_B}{\partial \mathbf{q}_B} \end{aligned} \tag{13}$$

where U_B is the potential energy of the body and $\mathbf{L}_B$ is the pseudo-inverse of $\mathbf{\Lambda}_B$ defined in Eq. (10) so that $\mathbf{L}_B\mathbf{\Lambda}_B = \mathbf{1}$. This wrench is treated as an intrinsic element of the system and is sometimes called the system wrench. On the other hand, the external wrench $\mathbf{w}_B^E$ is a resultant of forces and moments applied to the body from the external environments, which can be identified as specific objects. The kinematic constraint wrench $\mathbf{w}_B^K$ is essential to complete the dynamic equation of the individual body. However, it is non-working for the whole system and should be eliminated during the process of the global dynamics assembly. Furthermore, another term having the dimension of wrench appears in the global dynamics equations and it can be accounted for in the system wrench of the individual body. Therefore, the system wrench of the body is re-written as:

$$\mathbf{w}_B = -\mathbf{M}_B\dot{\mathbf{v}}_{B|\dot{q}=0} + \mathbf{w}_B^S \tag{14}$$

This is used as the representative state of the *wrench* object.

Joint

Joint is an object that abstracts the kinematics associated with the interconnections between the adjacent bodies. There are various types of interconnections,

and typical examples are the lower kinematic pairs such as the revolute, prismatic, universal, spherical types of joints, and so on. As a special case, an unconstrained motion of a body with respect to the base reference frame can be represented by a joint. The generic *joint* inherits most of the characters from the generic kinematic object and its state quantities are summarized as follows:

translation	rotation
$\bar{\mathbf{r}} = \mathbf{r}(\boldsymbol{\theta})$	$\bar{\mathbf{R}} = \mathbf{R}(\boldsymbol{\theta})$
$\overset{s}{\bar{\mathbf{r}}} = \mathbf{Z}_v \dot{\boldsymbol{\theta}}$	$\bar{\boldsymbol{\omega}} = \mathbf{Z}_\omega \dot{\boldsymbol{\theta}}$
$\overset{ss}{\bar{\mathbf{r}}}_{\|\dot{q}=0} = \dot{\mathbf{Z}}_v \dot{\boldsymbol{\theta}} + \mathbf{Z}_v \ddot{\boldsymbol{\theta}}_{\|\dot{q}=0}$	$\dot{\bar{\boldsymbol{\omega}}}_{\|\dot{q}=0} = \dot{\mathbf{Z}}_\omega \dot{\boldsymbol{\theta}} + \mathbf{Z}_\omega \ddot{\boldsymbol{\theta}}_{\|\dot{q}=0}$

In the above, $\boldsymbol{\theta}$ is the joint coordinate vector and $\mathbf{Z}_v$ and $\mathbf{Z}_\omega$ are the joint characteristic matrices for translation and rotation, respectively. All of these are specified by the type of the joint.

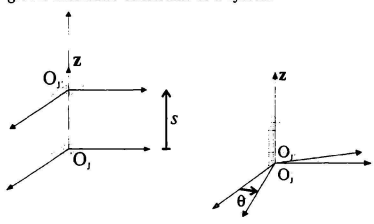
The generic kinematic object *joint* passes the motion states from one body to another. At the same time, it restricts the relative motion of a *frame* on one body with respect to a *frame* on an adjacent body. From the velocity transmission relations given by Eqs. (3 - 4), the kinematic constraint equation is

$$\mathbf{X}_J \begin{Bmatrix} \dot{\mathbf{p}}_J \\ \boldsymbol{\omega}_J \end{Bmatrix} + \mathbf{J}_J \dot{\boldsymbol{\theta}} = \begin{Bmatrix} \dot{\mathbf{p}}_{J'} \\ \boldsymbol{\omega}_{J'} \end{Bmatrix} \qquad (15)$$

where

$$\mathbf{X}_J = \begin{bmatrix} \bar{\mathbf{R}}^T & -\bar{\mathbf{R}}^T \bar{\mathbf{r}}^\times \\ \mathbf{0} & \bar{\mathbf{R}}^T \end{bmatrix} \quad \text{and} \quad \mathbf{J}_J = \begin{bmatrix} \bar{\mathbf{R}}^T \mathbf{Z}_v \\ \bar{\mathbf{R}}^T \mathbf{Z}_\omega \end{bmatrix}$$

are respectively the spatial transformation and local Jacobian matrices of the *joint*. As mentioned earlier, they play an important role in the assemblage of the global kinematic constraint of a system.



a) prismatic b) revolute

Fig. 3 The lower kinematic pairs

Prismatic and revolute are the most common types of joint in a wide range of mechanical applications, specially robotics. Both of them allow a single degree of freedom relative motion: translation in the prismatic type and rotation in the revolute type, which are depicted in Figures 3(a) and 3(b), respectively. The joint characteristic matrices $\mathbf{Z}_v$ and $\mathbf{Z}_\omega$ are simply related to the axis of the relative motion $\mathbf{z}$. They are summarized in the following:

prismatic		revolute	
$\bar{\mathbf{r}} = s \cdot \mathbf{z}$	$\bar{\mathbf{R}} = \mathbf{1}$	$\bar{\mathbf{r}} = \mathbf{0}$	$\bar{\mathbf{R}} = \mathbf{R}_z(\theta)$
$\mathbf{Z}_v = \mathbf{z}$	$\mathbf{Z}_w = \mathbf{0}$	$\mathbf{Z}_v = \mathbf{0}$	$\mathbf{Z}_w = \mathbf{z}$

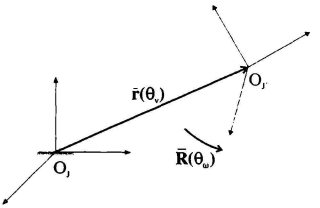


Fig. 4 Free-floating type joint

When a multibody system does not have a fixed base but is in a free-floating situation, the kinematic relation between the global reference frame and the first body of the multibody system can be abstracted as a *joint*, namely a *free-floating joint*. As depicted in Figure 4, it is certainly a kinematic connection between a frame of a body and that of its adjacent body if the inertial reference frame is treated as a body. This abstraction makes the proposed modeling concept consistent regardless of the condition of the base and any multibody system can be modeled similar to the case of a fixed base.

In a free-floating mode, the translation and rotation are not coupled to each other and can be considered separately. It is convenient to partition the joint coordinates into two parts so that $\boldsymbol{\theta} = [\boldsymbol{\theta}_v^T \quad \boldsymbol{\theta}_\omega^T]^T$. Accordingly, the joint characteristic matrices are partitioned and they are expressed as $\mathbf{Z}_v = [\mathbf{Z}_{vv} \quad \mathbf{0}]$ and $\mathbf{Z}_\omega = [\mathbf{0} \quad \mathbf{Z}_{\omega\omega}]$.

The translational motion is expressed in one of the three different coordinate systems depending on the nature of the motion and the corresponding states and parameters are listed in Table 1(**R.**-Rectangular, **C.**-Cylindrical, **S.**-Spherical).

It is preferable for the joint coordinates $\boldsymbol{\theta}$ to be independent, otherwise algebraic constraints should be taken into account in the system dynamics assembly. Hence, a choice from two sets of Euler angles is made

Table 1 Translation parameters

	$\boldsymbol{\theta}_v$	$\bar{\mathbf{r}}$	$\mathbf{Z}_{vv}$
R.	$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$	$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
C.	$\begin{bmatrix} r \\ \theta \\ z \end{bmatrix}$	$\begin{bmatrix} rC_\theta \\ rS_\theta \\ z \end{bmatrix}$	$\begin{bmatrix} C_\theta & -rS_\theta & 0 \\ S_\theta & rC_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$
S.	$\begin{bmatrix} r \\ \theta \\ \phi \end{bmatrix}$	$\begin{bmatrix} rC_\theta C_\phi \\ rS_\theta C_\phi \\ rS_\phi \end{bmatrix}$	$\begin{bmatrix} C_\theta C_\phi & -rS_\theta C_\phi & -rC_\theta S_\phi \\ S_\theta C_\phi & rC_\theta C_\phi & -rS_\theta S_\phi \\ S_\phi & 0 & rC_\phi \end{bmatrix}$

available to describe the rotational coordinates $\boldsymbol{\theta}_\omega$. Then. the kinematic states and parameters for the rotation are arranged in Table 2.

Table 2 Rotation parameters

variable	1-2-3 Euler	3-1-3 Euler
$\boldsymbol{\theta}_\omega$	$[\phi \quad \theta \quad \psi]^{\mathsf{T}}$	$[\phi \quad \theta \quad \psi]^{\mathsf{T}}$
$\bar{\mathbf{R}}$	$\mathbf{R}_x(\phi)\mathbf{R}_y(\theta)\mathbf{R}_z(\psi)$	$\mathbf{R}_z(\phi)\mathbf{R}_x(\theta)\mathbf{R}_z(\psi)$
$\mathbf{Z}_{\omega\omega}$	$\begin{bmatrix} 1 & 0 & S_\theta \\ 0 & C_\phi & -S_\phi C_\theta \\ 0 & C_\phi & C_\phi C_\theta \end{bmatrix}$	$\begin{bmatrix} 0 & C_\phi & S_\phi S_\theta \\ 0 & S_\phi & -C_\phi S_\theta \\ 1 & 0 & C_\theta \end{bmatrix}$

In Tables 1 and 2, $C_{(\cdot)}$ and $S_{(\cdot)}$ are the abbreviations of $\cos(\cdot)$ and $\sin(\cdot)$, respectively. $\mathbf{R}_\alpha(\beta)$ represent the rotation matrix associated with the rotation of the amount of β about the axis α.

Multibody system

A multibody system is composed of several bodies and joints in a certain topological configuration. Its dynamics modeling could be achieved by the organization of only the body and joint objects and the relationship between them, if these objects were as fully autonomous as the real world objects. This would be possible if each object were equipped with its own processor. In practical modeling, however, a *system* object is introduced and it is in charge of the organization of the body and joint objects. It generates the governing equations of motion of the multibody system by assembling the kinematic and dynamic states of the body and joint objects. The assembly operation of the *system* is based on the dynamics formulation using the natural orthogonal complement of the kinematic constraint matrix.

One of the goals of this paper is to consider multibody systems having a tree-type topology. The identification of the arrangement of the members in a multibody system is accomplished by the lower body

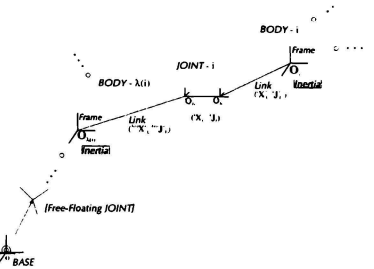


Fig. 5 Connected bodies

index representation:[9] (i) A base is identified in the system; (ii) A number is assigned to each body as an index, usually in ascending progression away from the base; (iii) For a body of index i, $\lambda(i)$ is assigned as the index number of the lower body that is contiguous to the body of index i and lies on the way to the base.

The motion of a connected body is constrained with respect to that of the corresponding lower body. It is a kinematic constraint due to the combination of the outward *link* in the lower body. the joint between them, and the inward *link* in the body as shown in Figure 5. In terms of the aforementioned spatial transformation and local Jacobian matrices, the individual constraint equations are respectively

$$ {}^{\lambda(i)}\mathbf{X}_L^{\circ} \begin{Bmatrix} \dot{\mathbf{p}}_{\lambda(i)} \\ \boldsymbol{\omega}_{\lambda(i)} \end{Bmatrix} + {}^{\lambda(i)}\mathbf{J}_L^{\circ}\, \dot{\mathbf{u}}_{\lambda(i)} = \begin{Bmatrix} \dot{\mathbf{p}}_{J_i} \\ \boldsymbol{\omega}_{J_i} \end{Bmatrix} \tag{16} $$

$$ {}^{i}\mathbf{X}_J \begin{Bmatrix} \dot{\mathbf{p}}_{J_i} \\ \boldsymbol{\omega}_{J_i} \end{Bmatrix} + {}^{i}\mathbf{J}_J\, \dot{\boldsymbol{\theta}}_i = \begin{Bmatrix} \dot{\mathbf{p}}_{J_i'} \\ \boldsymbol{\omega}_{J_i'} \end{Bmatrix} \tag{17} $$

$$ {}^{i}\mathbf{X}_L' \begin{Bmatrix} \dot{\mathbf{p}}_{J_i'} \\ \boldsymbol{\omega}_{J_i'} \end{Bmatrix} + {}^{i}\mathbf{J}_L'\, \dot{\mathbf{u}}_i = \begin{Bmatrix} \dot{\mathbf{p}}_i \\ \boldsymbol{\omega}_i \end{Bmatrix} \tag{18} $$

The super/sub-script indices are attached for member identification. The substitution of Eqs. (16) and (17) into Eq. (18) results in:

$$ {}^{i}\mathbf{X}_L'\, {}^{i}\mathbf{X}_J\, {}^{\lambda(i)}\mathbf{X}_L^{\circ} \begin{Bmatrix} \dot{\mathbf{p}}_{\lambda(i)} \\ \boldsymbol{\omega}_{\lambda(i)} \end{Bmatrix} + {}^{i}\mathbf{X}_L'\, {}^{i}\mathbf{X}_J\, {}^{\lambda(i)}\mathbf{J}_L^{\circ}\, \dot{\mathbf{u}}_{\lambda(i)} + {}^{i}\mathbf{X}_L'\, {}^{i}\mathbf{J}_J\, \dot{\boldsymbol{\theta}}_i + {}^{i}\mathbf{J}_L'\, \dot{\mathbf{u}}_i = \begin{Bmatrix} \dot{\mathbf{p}}_i \\ \boldsymbol{\omega}_i \end{Bmatrix} \tag{19} $$

which is the kinematic constraint equation between body i and the lower body $\lambda(i)$.

One of the useful generalized coordinates in the tree-type topology is a serial combination of the joint coordinates and the modal coordinates of body so that $\mathbf{q} = [\cdots \boldsymbol{\theta}_i^{\mathsf{T}}\ \mathbf{u}_i^{\mathsf{T}} \cdots]^{\mathsf{T}}$. Then, there exist mappings from the global to the local, such as $\dot{\boldsymbol{\theta}}_i = \mathbb{C}_{\theta_i}\dot{\mathbf{q}}$, $\dot{\mathbf{u}}_i = \mathbb{C}_{u_i}\dot{\mathbf{q}}$, and $\{\dot{\mathbf{p}}_i^{\mathsf{T}}\ \boldsymbol{\omega}_i^{\mathsf{T}}\}^{\mathsf{T}} = \mathbb{V}_i\dot{\mathbf{q}}$. The first two mappings are straightforward from the definition of $\mathbf{q}$ and $\mathbb{C}_{\theta_i}$ and

$\mathbb{C}_{u_i}$ are connectivity matrices whose elements are simply 0 and 1. Because of the independence of the generalized coordinates $\mathbf{q}$ in the tree-type topology, the introduction of these mappings into Eq. (19) results in:

$$ {}^{I}\mathbf{X}'_{L}\,{}^{I}\mathbf{X}_{J}{}^{\lambda(i)}\mathbf{X}^{o}_{L}\,\mathbb{V}_{\lambda(i)} + {}^{I}\mathbf{X}'_{L}\,{}^{I}\mathbf{X}_{J}{}^{\lambda(i)}\mathbf{J}^{o}_{L}\,\mathbb{C}_{u_{\lambda(i)}} + {}^{I}\mathbf{X}'_{L}\,{}^{I}\mathbf{J}_{J}\,\mathbb{C}_{\theta_i} + {}^{I}\mathbf{J}'_{L}\,\mathbb{C}_{u_i} = \mathbb{V}_i, \qquad i > 1 \quad (20) $$

$$ {}^{1}\mathbf{X}'_{L}\,{}^{1}\mathbf{J}_{J}\,\mathbb{C}_{\theta_1} + {}^{1}\mathbf{J}'_{L}\,\mathbb{C}_{u_1} = \mathbb{V}_1 \quad (21) $$

In the above, Eq. (21) is just an application of Eq. (20) for a set of the first joint and body connected to the base reference frame. Eqs. (20) and (21) constitute a recursion formula that evaluates the local velocity mapping matrices $\mathbb{V}_i$ from the base to the terminal bodies of the system.

In terms of the local velocity mapping matrices $\mathbb{V}_i$ and the connectivity matrices $\mathbb{C}_{u_i}$ for the modal coordinates, the natural orthogonal complement matrix of the kinematic constraints is obtained as:

$$ \mathbf{N} = \left[\cdots \mathbb{V}_i^{\mathrm{T}}\,\mathbb{C}_{u_i}^{\mathrm{T}} \cdots \right]^{\mathrm{T}} \quad (22) $$

It maps the generalized speed $\dot{\mathbf{q}}$ into the global extended velocity vector $\mathbf{v}$, namely, $\mathbf{v} = \mathbf{N}\dot{\mathbf{q}}$. Using the principle of virtual work and the fact that the global kinematic constraint wrench $\mathbf{w}^K$ is non-working on the system, the natural orthogonal complement matrix $\mathbf{N}$ can be shown to be orthogonal to $\mathbf{w}^K$, i.e., $\mathbf{N}^{\mathrm{T}}\mathbf{w}^K = \mathbf{0}$. These features of the natural orthogonal complement matrix are useful to transform the aggregate of the equations of motion of the individual bodies, Eq. (11), into the governing equation of motion of the system as follows:

$$ \left(\mathbf{N}^{\mathrm{T}}\mathbf{M}\mathbf{N}\right)\ddot{\mathbf{q}} = -\mathbf{N}^{\mathrm{T}}\mathbf{M}\dot{\mathbf{N}}\dot{\mathbf{q}} + \mathbf{N}^{\mathrm{T}}\left(\mathbf{w}^S + \mathbf{w}^E\right) \quad (23) $$

In the above, $\mathbf{M}$ is a block diagonal matrix whose elements are the mass matrices $\mathbf{M}_B$ of individual bodies, while $\mathbf{w}^S$ and $\mathbf{w}^E$ are the stacks of the individual system and external wrenches, respectively. It should be mentioned that the global kinematic constraint wrench has been eliminated from Eq. (23) by the premultiplication of $\mathbf{N}^{\mathrm{T}}$, because $\mathbf{N}^{\mathrm{T}}\mathbf{w}^K = \mathbf{0}$. Since the evaluation of $\dot{\mathbf{N}}$ involves a large number of complications, $\dot{\mathbf{N}}\dot{\mathbf{q}} = \dot{\mathbf{v}}_{|\dot{q}=0}$ replaces it in Eq. (23). Actually, $\dot{\mathbf{v}}_{|\dot{q}=0}$ are evaluated by the *frames* of the individual bodies of the system. As a result, the dynamic equation of motion of the system is obtained in terms of the minimal coordinates as follows:

$$ \mathbf{I}(\mathbf{q})\,\ddot{\mathbf{q}} = \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) + \boldsymbol{\tau} \quad (24) $$

where

$$ \begin{aligned} \mathbf{I} &= \mathbf{N}^{\mathrm{T}}\mathbf{M}\mathbf{N} \\ \mathbf{c} &= \mathbf{N}^{\mathrm{T}}\left(-\mathbf{M}\dot{\mathbf{v}}_{|\dot{q}=0} + \mathbf{w}^S\right) \\ \boldsymbol{\tau} &= \mathbf{N}^{\mathrm{T}}\mathbf{w}^E \end{aligned} $$

are respectively the generalized inertia matrix, convective inertial force vector, and generalized external force vector of the system. In addition to the generalized coordinates and their time derivatives, they are registered as the representative states of the *system* object.

Implementation

The objects mentioned in the previous section are refined from the generic types such as kinematic and dynamic in a top-down fashion. Each object is endowed with its own characteristic quantities and functionality, which are evolved from those of the generic type by taking advantage of the inheritance concept. Most of the objects have a relationship with others, then the boundaries and interfaces of the objects are carefully designed. The elementary objects for modeling dynamics of rigid/flexible multibody systems are identified and listed in Table 3.

In addition to the elementary objects, several supplementary objects are introduced in order to complete the simulation needs. These are *solver*, *damping*, *joint activations*, *signals*, and so on.

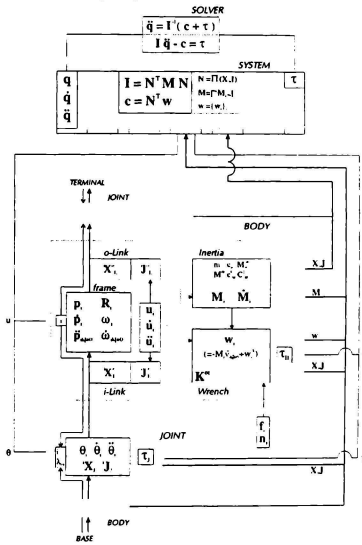


Fig. 6 Modeling scheme

The basic modeling scheme is shown in Figure 6. A multibody simulation model is a combination of bodies, *joints*, a *system*, and a *solver*, while each body in

Table 3 Multibody objects

Class	Object	Characteristic quantities	Companions
kinematics	frame	position($\mathbf{p}_B$, $\mathbf{R}_B$) velocity($\dot{\mathbf{p}}_B$, $\boldsymbol{\omega}_B$) acceleration($\ddot{\mathbf{p}}_{B\|\dot{q}=0}$, $\dot{\boldsymbol{\omega}}_{B\|\dot{q}=0}$)	links, joints, dynamics
	modal	coordinates($\mathbf{u}_B$,$\dot{\mathbf{u}}_B$,$\ddot{\mathbf{u}}_B$)	links, dynamics, system
	rigid link (in, out)	length($\bar{\mathbf{r}}$), twist($\bar{\mathbf{R}}$) SpTr($\mathbf{X}_L$)	frame, joints
	flexible link (in, out)	length($\bar{\mathbf{r}}$), twist($\bar{\mathbf{R}}$) SpTr($\mathbf{X}_L$), L.Jacobian($\mathbf{J}_L$) mode shapes(${}^B\boldsymbol{\Phi}$, ${}^B\boldsymbol{\Gamma}$)	frame, modal, joints
	prismatic joint	stretch(s, $\dot{s}$, $\ddot{s}$) parameters($\mathbf{Z}_v$, $\mathbf{Z}_\omega$) SpTr($\mathbf{X}_J$), L.Jacobian($\mathbf{J}_J$)	frame, base, links system
	revolute joint	rotation(θ, $\dot{\theta}$, $\ddot{\theta}$) parameters($\mathbf{Z}_v$, $\mathbf{Z}_\omega$) SpTr($\mathbf{X}_J$), L.Jacobian($\mathbf{J}_J$)	frame, base, links system
	free-floating (joint)	coordinates($\boldsymbol{\theta}$, $\dot{\boldsymbol{\theta}}$, $\ddot{\boldsymbol{\theta}}$) parameters($\mathbf{Z}_v$, $\mathbf{Z}_\omega$) SpTr($\mathbf{X}_J$), L.Jacobian($\mathbf{J}_J$)	frame, base, links system
dynamics	inertia	mass matrix($\mathbf{M}_B$, $\mathbf{M}_B$)	frame, modal, wrench system
	wrench	system wrench($\mathbf{w}_B$) external wrench($\mathbf{w}_B^E$)	frame, modal, inertia system
system	forward	lowerbody array($\lambda(i)$) coordinates($\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$), torque($\boldsymbol{\tau}$) inertia($\mathbf{I}$), convective force($\mathbf{c}$)	modal, joints, dynamics
	inverse	lowerbody array($\lambda(i)$) torque($\boldsymbol{\tau}$), coordinates($\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$) inertia($\mathbf{I}$), convective force($\mathbf{c}$)	modal, joints, dynamics

turn is composed of a *frame*, a *modal*, inward/outward *links*, an *inertia*, and a *wrench*.

The *frames* that hold the motion states of bodies are updated in a recursive manner from the *base* to the terminal bodies of the system through the *link* and *joint* operations. The *inertia* and *wrench* objects in each body refresh themselves based on the *frame* of the body. The bodies export the spatial transformation and local Jacobian matrices of *links* as the representative kinematics and the mass matrix and wrench vector as the representative dynamics to the *system*, while the *joints* do the kinematics only.

In terms of the spatial transformation and local Jacobian matrices gathered from the bodies and joints, the *system* constructs the natural orthogonal complement of the kinematic constraint matrix. Then, it evaluates the generalized inertia matrix and convective force vector using the mass matrices and wrench vectors from the bodies.

The *solver* calculates the generalized force vector with the input of the generalized coordinates and their time derivatives of the *system* for the inverse dynamics case, while it integrates the equations of motion of the *system* for the forward dynamics case.

The aforementioned objects are implemented as class modules in ROSE(Real-time Object-oriented Simulation Environment). The objects are instantiated with graphic icons and an actual modeling is achieved by the interconnection of these object icons in a simple and intuitive manner. An example of modeling in ROSE is shown in Figure 7.

Validation

The scope of the present paper is to model and simulate the dynamics of tree-type rigid/flexible multibody systems in a free-floating mode, which are common in many space applications such as a satellite with deployable appendages or a space manipulator system. For the validation purpose, relatively simple configurations are chosen from the said category. They are simultaneously modeled by the proposed object-oriented modeling and any others, then their simulation results are compared.

The first configuration considered is composed of

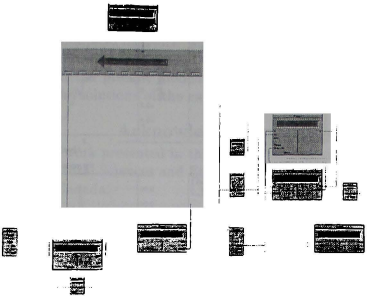


Fig. 7 Modeling in ROSE

three rigid slender bodies and two revolute joints. As depicted in Figure 8, the origins of the two revolute joints coincide at one point so that the topology of the system becomes tree-type. Neither of the bodies is constrained with respect to the inertial reference frame and the system is in a free-floating mode. The physical parameters are arbitrarily chosen and listed in Table 4. The parameters of each body are resolved with respect to the body fixed reference frame whose origin is at the left tip and x-axis is along the longitudinal axis of the body.

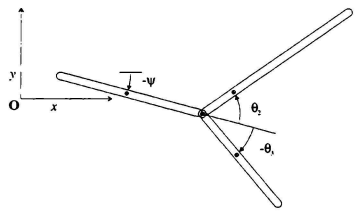
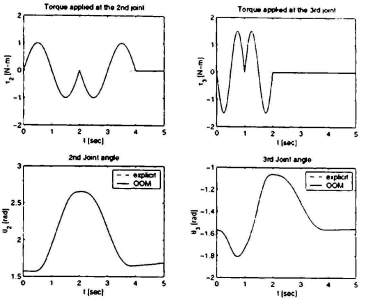


Fig. 8 Free-Floating three-bar system

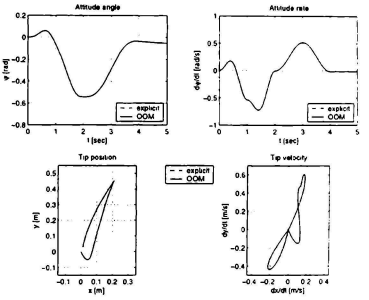
Table 4 The parameters of the three-bar system

body	length [m]	mass [kg]	C.M. [m]	I_y [kg·m^2]	I_z [kg·m^2]
1	1.0	1.0	0.5	1.0	1.0
2	1.0	1.0	0.5	1.0	1.0
3	1.0	1.0	0.5	1.0	1.0

The corresponding object-oriented model is composed of three *body* classes and three *joint* objects, where the first joint is of the *free-floating* type and connects the inertial frame to the first body. In the case of planar motion, the equations of motion can be obtained in an explicit form and can be solved without difficulty by a commercial software such as MATLAB. When the given torques are applied at the two joints, the responses of the attitude of the first body as well as the joints are compared between the proposed object-oriented model and the explicit formulation. Figure 9(a) shows the time histories of the joint angles while Figure 9(b) presents the time histories of the attitude angle and its rate as well as the tip position and velocity profile. There is very good agreement between the two sets of results.



a) the comparison of the joint response



b) the comparison of the attitude response

Fig. 9 The response of the three-bar system

A rigid-flexible manipulator in a free-floating mode is considered next as an another validation model. As depicted in Figure 10, it is a combination of a rigid bar and a uniform beam, which are connected by a revolute joint. Because of the uniform beam, any practical modeling process for numerical simulation should involve a certain kind of discretization. In the present paper the assumed mode method is adopted and the beam functions for the clamped-free boundary condi-

tion are used as the mode shapes. Then, the equations of motion of the system are available in an explicit form and can be solved by a mathematical software. In the corresponding object-oriented model, a free-floating joint is added in between the inertial reference frame and the rigid bar. The physical parameters of the system are arbitrarily chosen for the numerical validation purpose and are listed in Table 5.

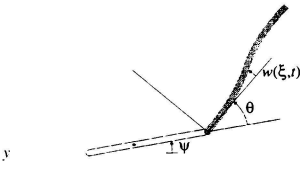


Fig. 10 Rigid-flexible manipulator

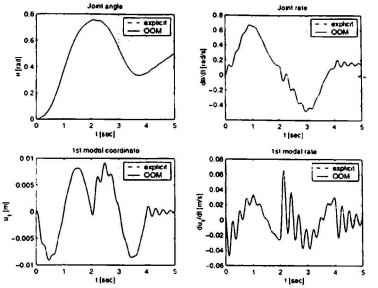
Table 5 The parameters of the rigid-flexible manipulator

body	length [m]	mass [kg]	I_y [kg·m^2]	I_z [kg·m^2]	EI [N·m^2]
1	1.0	1.0	0.3333	0.3333	-
2	1.0	1.0	0.3333	0.3333	1.0

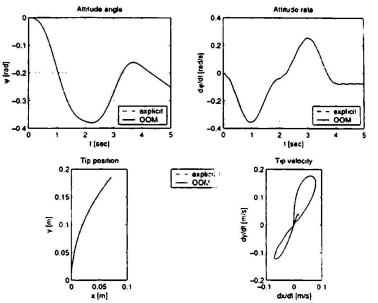
The same pattern of the torque given at the second joint in the previous validation case is applied with the amplitude 0.05 [Nm] at the joint of the rigid-flexible manipulator model. The corresponding responses obtained from the object-oriented simulation and the solution of the explicit formulation are compared. As shown in Figure 11(a), the joint and modal responses are in good agreement. In the simulation the structural damping is assumed to be 1 % of the critical damping, and its effect can be identified in the response. The attitude response obtained from the two simulations are in good coincidence, in fact they are hardly distinguishable in Figure 11(b).

Conclusions

Multibody systems are composed of many components, each of which can be abstracted into a mathematical modeling unit called object by the assignment of the characteristic variables and operations. Groups of objects have similar internal structure and they are spawned from generic classes. An object-oriented modeling of a multibody system can be achieved by the interconnection of the pre-casted object modules in a simple manner. It has a lower chance of error and it is amenable to a fast modification of an already developed model.



a) the comparison of the joint and modal response



b) the comparison of the attitude and tip motion

Fig. 11 The response of the rigid-flexible manipulator

The material flexibility of bodies is intimately involved in the representations of both the kinematics and dynamics of the system. The modal representation is useful for the former and the energy expression conveniently accommodates the body flexibility into the latter. The Lagrange formulation in conjunction with the natural orthogonal complement of kinematic constraint matrix successfully works for the object-oriented scheme, because it maintains a modular representation for the kinematics and dynamics of an individual body in a system.

When a multibody system is in a free-floating mode, the kinematic relationship between the inertial reference base and the first body of the system is abstracted as a *joint* type. This is useful and introduces a consistent and uniform modeling concept in which any multibody system can be treated as if it were originated from the inertial reference base.

The proposed object-oriented modeling approach

was validated by simulation of the dynamics of several examples, in which the equations of motion can be obtained in an explicit form. The simulation results of the present modeling are in good agreement with the solutions of the explicit formulations.

Acknowledgement

The work presented in this paper was supported by the Natural Sciences and Engineering Research Council of Canada.

References

[1]Booch, G., *Object-oriented analysis and design with applications*, Benjamin-Cummings Pub., 1994.

[2]Otter, M., Hocke, M., Daberkow, A., and Leister, G., "An object-oriented data model for multibody systems," *Advanced Multibody System Dynamics*, edited by W. Schiehlen, Kluwer Academic Publishers, 1993, pp. 19 – 48.

[3]Wallrapp, O., "Standard input data of flexible members in multibody systems." *Advanced Multibody System Dynamics*, Kluwer Academic Publishers, 1993, pp. 445 – 450.

[4]Wallrapp, O., "Standardization of flexible body modeling in multibody system codes, Part 1: Definition of standard input data." *Mechanics of Structures and Machines*, Vol. 22, No. 3, 1994, pp. 283 - 304.

[5]Lückel, J., Junker, F., and Toepper, S., "Block-oriented modelling of rigid multibody systems with regard to subsystem techniques," *Advanced Multibody System Dynamics*, edited by W. Schiehlen, Kluwer Academic Publishers, 1993, pp. 49 – 66.

[6]Kecskemethy, A., "Object-oriented modeling of mechanical systems," *Kinematics and Dynamics of Multibody Systems*, edited by J. Angeles and A. Kecskemethy, chap. 6, Springer-Verlag, 1995, pp. 217 – 276.

[7]Anantharaman, M., "Flexible multibody dynamics - an object-oriented approach," *Nonlinear Dynamics*, Vol. 9, No. 1 - 2, 1996, pp. 205 – 221.

[8]Cyril, X., Angeles, J., and Misra, A., "Dynamics of flexible multibody mechanical systems," *Transactions of the CSME*, Vol. 15, No. 3, 1991, pp. 235 – 256.

[9]Huston, R., *Multibody Dynamics*, Butterworths, 1990.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37237

AIAA-2000-4178

A NEW METHOD TO OBTAIN A UNIQUE SOLUTION IN SYSTEM IDENTIFICATION

Masayoshi Misawa*
Shizuoka University
Hamamatsu, Shizuoka, Japan

Abstract

This paper deals with the uniqueness of the identified mass and stiffness matrices. Using the constrained minimization theory, these matrices are identified so that the Euclidean norm of the matrices can be minimized. Mass matrix, identified by using the mode orthogonality conditions as constraints, is not unique. Measured mass properties of a structure are introduced into the constraints to solve this problem. A numerical example is given to demonstrate that the proposed method is effective for system identification. It is found that the identified mass and stiffness matrices can provide a highly accurate finite-element model mating with both the vibration test and the mass property test results.

Introduction

Stiffness is one of the most important design requirements for space structures. Space structures in orbit have to meet specified frequency requirements to avoid coupling with both their control systems and other equipment. Structural designs to satisfy these requirements are relatively easy for small structures. However, it becomes difficult to meet the frequency requirements for large structures. Natural frequencies decrease as the size of a space structure increases, and they come close to the specified requirements. Therefore, we need more accurate predictions to confirm the frequency margin during the design phase. Finite element analysis is a powerful tool to obtain dynamic characteristics for space structures. Finite-element models of those structures will need a large number of degrees of freedom as they become larger. In such cases, it is very difficult to find accurate solutions due to modeling errors of initial finite-element models.

System identification is a key technology in predicting the accurate dynamic behavior of structures. Several types of methods[1-6] have been proposed to establish the analytical models mating with modal test results. One of these methods assumes an initial, undamped, finite-element model which is characterized by its mass and stiffness matrices[3-6]. The mass and stiffness matrices are corrected by minimizing the Euclidean norm subject to constraints. These constraints include mode orthogonality, the symmetry of mass and stiffness matrices, and the connectivity of finite-elements as proposed by Kabe[7].

First of all, this paper describes some problems in current mass matrix identification. One problem is that the identified mass matrix is not unique and the other is that we cannot obtain mass properties of real structures from the identified mass matrix. This means mass distribution of the analytical model is different from that of the real structures. Next, the cause of these problems is clarified. The cause is the use of mode shapes in the constraints. Because the mode shape is not uniquely defined, any mode shape can be multiplied by a nonzero constant without changing its physical meaning. As a result, there exists an infinite number of identified mass matrices that satisfy the constraints. Baruch[8] also points out the same problem when all modes are considered (the modal matrix is square). Next, a new method is proposed to solve these problems. Finally, a numerical example is given to demonstrate that the proposed method is effective for system identification.

Nomenclature

$\boldsymbol{A}$	: Diagonal matrix with arbitrary constant as its elements
$\boldsymbol{B}$	: Matrix defined by Eq. (38)
$\boldsymbol{E}$	: Unit matrix
f	: Natural frequency
$\boldsymbol{G}$	: Matrix defined by Eq. (31)
$\boldsymbol{H}$	: Matrix defined by Eq. (34)
I_x, I_y, I_z	: Measured moment of inertia of structures
$\boldsymbol{K}$	: Identified stiffness matrix
$\boldsymbol{K}_A$	: Stiffness matrix of finite-element model
$\Delta\boldsymbol{K}$	: Difference between $\boldsymbol{K}$ and $\boldsymbol{K}_A$
$\boldsymbol{L}$	: Matrix defined by Eq. (29)
$\boldsymbol{M}$	: Identified mass matrix
$\boldsymbol{M}_A$	: Mass matrix of finite-element model

*Professor, Department of Mechanical Engineering. Senior member of AIAA

Copyright © 2000 by the American Institute of Aeronautic and Astronautics, Inc. All rights reserved.

M_0 : Rigid-body mass matrix defined by Eq. (16)
ΔM : Difference between M and M_A
m : Measured mass of structure
n_A : Total number of analytical modes
n_T : Total number of measured modes
P : Matrix defined by Eq. (3)
R : Rigid-body modes
T : Matrix defined by Eq. (37)
W : Matrix defined by Eq. (35)
x_G, y_G, z_G : Measured center of gravity of structures
α, β, γ : Lagrange multiplier
ϕ : Modal matrix
$\bar{\phi}$: Modal matrix defined by Eq. (8)
Ω^2 : Measured frequency matrix
ω_i : Measured circular frequencies of i-th mode
ψ : Lagrange function
Λ : Matrix with Lagrange multiplier α_{ij} as its elements

The Current Method In Mass Matrix Identification

A typical method for system identification is described below. This method has the potential to update finite element models with large degrees of freedom. In this method, the modal matrix satisfies the following orthogonality condition:

$$\phi^T(M_A + \Delta M)\phi = E \tag{1}$$

Equation (1) can be rewritten as

$$\phi^T \Delta M \phi = E - P \tag{2}$$

where

$$P = \phi^T M_A \phi \tag{3}$$

Mass matrix is identified by minimizing the Euclidean norm $\|M - M_A\|$ subject to the mode orthogonality conditions. Using the constrained minimization theory, the Lagrange function is written by

$$\psi = \|M_A^{-1/2} \Delta M M_A^{-1/2}\| / 2 + \sum_{i=1}^{n_t}\sum_{j=1}^{n_A} \beta_{ij}(M - M^T)_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_{ij}(\phi^T \Delta M \phi - E + P)_{ij} \tag{4}$$

Differentiating Eq. (4) with respect to each element of ΔM and equating the results to zero gives Eq. (5) that ΔM must satisfy when ψ is minimum.

$$\Delta M = -\frac{1}{2} M_A \phi \Lambda^T \phi^T M_A \tag{5}$$

The following equation can be obtained by substituting Eq. (5) into Eq. (2).

$$\Lambda = -2P^{-1}(E - P)P^{-1} \tag{6}$$

By substituting Eq. (6) into Eq. (5), mass matrix can be identified as

$$M = M_A + M_A \phi P^{-1}(E - P) P^{-1} \phi^T M_A \tag{7}$$

Problems and Their Causes In the Current Methods

One problem is that the identified mass matrix is not unique. The cause of this problem is the use of mode shapes in the constraints. Because the mode shape is not uniquely defined, any mode shape can be multiplied by a nonzero constant without changing its physical meaning.

$$\bar{\phi} = \phi A \tag{8}$$

If we assume that another solution $\Delta \bar{M}$ exists satisfying Eq. (2), the following equation is obtained.

$$\bar{\phi}^T \Delta \bar{M} \bar{\phi} = E - \bar{P} \tag{9}$$

where

$$\bar{P} = \bar{\phi}^T M_A \bar{\phi} \tag{10}$$

Substitution of Eq. (8) into Eq. (10) gives

$$\bar{P} = A P A \tag{11}$$

In the same way as stated above, matrix $\Delta \bar{M}$ and matrix $\bar{\Lambda}$ are obtained as

$$\Delta \bar{M} = -\frac{1}{2} M_A \bar{\phi} \bar{\Lambda}^T \bar{\phi}^T M_A \tag{12}$$

$$\bar{\Lambda} = -2A^{-1}P^{-1}(A^{-2} - P) P^{-1} A^{-1} \tag{13}$$

Therefore, the identified mass matrix is expressed as

$$M = M_A + M_A \phi P^{-1}(A^{-2} - P) P^{-1} \phi^T M_A \tag{14}$$

This equation shows that there exist infinite identified mass matrices that satisfy the mode orthogonality conditions because all the elements of the diagonal matrix A are arbitrary. Namely, the identified mass matrix is not unique. It should be noted that the mode shape is the ratio of amplitude at each point on a structure. Therefore, we can normalize the modes in several different expressions according to our purpose. Equation (1) is just one of these expressions. The mode orthogonality condition is not the constraint imposed on mass matrices but on mode shapes. It is not reasonable to use the mode orthogonality condition for mass matrix identification. This is the reason the identified mass matrix is not unique.

The other problem is that the identified mass matrix cannot provide mass properties of structures mating with the test results. Only several measured modes are required to identify the mass matrix, as shown in Eq. (7). It is especially difficult to determine mass distribution of the structure by using modes. This means it cannot be assured that the identified mass matrix approaches the true mass matrix of real structures. As a result, the identified mass matrix gives physically unreason-

able results, which state that the weight of the structure has different values in different directions, as described later in numerical example.

On the other hand, the identified stiffness matrix is unique even though any normalized mode is used in its identification. It can be understood that the uniqueness of mass matrix depends on the constraints as shown in Eq. (14). Therefore, we confirm the uniqueness of the stiffness matrix by checking the constrains. The symmetry requirement of the stiffness matrix, mode orthogonality conditions and the dynamic equation are selected as constraints.

The modal matrix satisfies the dynamic equation:

$$K\phi = M\phi\Omega^2 \tag{15}$$

Premultiplying Eq. (15) by ϕ^T, we obtain the orthogonality conditions:

$$\phi^T K\phi = \phi^T M\phi\,\Omega^2 \tag{16}$$

It should be noted that modal masses are not normalized to be one. If another solution $\bar{K}$ exists which satisfies the dynamic equation, we obtain

$$\bar{K}\bar{\phi} = M\bar{\phi}\Omega^2 \tag{17}$$

The following equation is obtained by substituting Eq. (8) into Eq. (17).

$$\bar{K}\phi A = M\phi A\Omega^2 \tag{18}$$

Considering that both matrices Ω^2 and A are diagonal matrices, Eq. (18) is rewritten as

$$\bar{K}\phi A = M\phi\Omega^2 A \tag{19}$$

Postmultiplying Eq. (19) by A^{-1} gives

$$\bar{K}\phi = M\phi\Omega^2 \tag{20}$$

From Eqs. (15) and (20), we obtain

$$\bar{K} = K \tag{21}$$

Therefore, mode normalization gives no effect on the constraint of the dynamic equation.

We can reach the same result for the mode orthogonality conditions. In a similar way, another solution $\bar{K}$ satisfies the following equation.

$$\bar{\phi}^T\bar{K}\bar{\phi} = \bar{\phi}^T M\bar{\phi}\,\Omega^2 \tag{22}$$

Substitution of Eq. (8) into Eq. (16) gives

$$\begin{aligned} A^T\phi^T\bar{K}\phi A &= A^T\phi^T M\phi A\,\Omega^2 \\ &= A^T\phi^T M\phi\,\Omega^2 A \end{aligned} \tag{23}$$

Premultiplying and post-multiplying Eq. (23) by A^{-1}, we obtain

$$\phi^T\bar{K}\phi = \phi^T M\phi\,\Omega^2 \tag{24}$$

This shows the identified stiffness matrix is always unique if the identified mass matrix is unique.

Proposed Method

Flexible space structures, such as antennas and solar paddles, are subjected to vibration and mass property tests to verify their performance. It is reasonable to correct analytical models based on discrepances between the measured and predicted results concerning modal characteristics and mass property. We introduced the mass properties of structures into constraint to solve the above problems.

It is essentially impossible to get an accurate mass matrix by update using mode shapes only. Mass matrix is assembled using geometry data and material data of the finite-elements. Mass matrix has the potential to express the mass property of the real structures if the modeling error is small. However, mass property data are not utilized in the current methods of mass matrix identification. It is reasonable to introduce the mass properties into the constraints from a physical point of view. Rigid-body mass matrix, calculated from the mass matrix and rigid-body modes, gives the mass property information of the structures. Each element of the rigid-body mass matrix is unique. Therefore, the proposed method can identify a unique mass matrix and can provide mass properties which are identical to the measured data.

Rigid-body mass matrix, whose elements show mass properties, is given by $R^T M R$. Mass matrix must satisfy the following relation

$$R^T M R = M_0 \tag{25}$$

where M_0 is a matrix with measured data as its element and is expressed as

$$M_0 = \begin{bmatrix} m & & & & & \\ 0 & m & & & sym. & \\ 0 & 0 & m & & & \\ 0 & -z_G m & y_G m & I_x & & \\ z_G m & 0 & -x_G m & I_{xy} & I_y & \\ -y_G m & x_G m & 0 & I_{xz} & I_{yz} & I_z \end{bmatrix} \tag{26}$$

The Lagrange function is written by

$$\begin{aligned} \psi = \| \Delta M \|/2 &+ \sum_{i=1}^{n_A}\sum_{j=1}^{n_A} \beta_{ij}(M - M^T)_{ij} \\ &+ 2\sum_{i=1}^{n_T}\sum_{j=1}^{n_T} \gamma_{ij}(R^T M R - M_0)_{ij} \end{aligned} \tag{27}$$

Differentiating Eq. (27) with respect to each element of ΔM and equating the results to zero gives Eq. (28) that ΔM must satisfy when ψ is minimum.

$$M = M_A - \sum_{i=1}^{\hat{6}}\sum_{j=1}^{\hat{6}} \gamma_{ij}(L_{ij} + L_{ij}^T) \tag{28}$$

where

$$L_{ij} = < R_i\, R_j^T > \tag{29}$$

and < * > shows a matrix that has the same structural connectivity as the original mass matrix. That is, matrix < * > has a value of zero at all degrees of freedom with values of zero in the original matrix. Taking into account the mass property constraints, one obtains

$$\sum_{i=1}^{6}\sum_{j=1}^{6}\gamma_{ij}\boldsymbol{G}_{ij} = \boldsymbol{R}^T\boldsymbol{M}_A\boldsymbol{R} - \boldsymbol{M}_D \tag{30}$$

where

$$\boldsymbol{G}_{ij} = \boldsymbol{R}^T(\boldsymbol{L}_{ij} + \boldsymbol{L}_{ij}^T)\boldsymbol{R} \tag{31}$$

Mass matrix is identified by substituting the Lagrange multipliers obtained from Eq. (30) into Eq. (28).

The Stiffness matrix can be identified in a similar way. The Lagrange function is written by

$$\begin{aligned}\psi &= \|\Delta\boldsymbol{K}\|/2 + \sum_{i=1}^{n}\sum_{j=1}^{n_A}\pi_{ij}(\boldsymbol{K}-\boldsymbol{K}^T)_{ij} \\ &+ 2\sum_{i=1}^{n_T}\sum_{i=1}^{n_T}\nu_{ij}(\boldsymbol{\phi}^T\boldsymbol{K}\boldsymbol{\phi} - \boldsymbol{\phi}^T\boldsymbol{M}\boldsymbol{\phi}\boldsymbol{\Omega}^2)_{ij} \\ &+ 2\sum_{k=1}^{n_T}\sum_{j=1}^{n_A}\lambda_{ij}(\boldsymbol{K}\boldsymbol{\phi} - \boldsymbol{M}\boldsymbol{\phi}\boldsymbol{\Omega}^2)_{ij}\end{aligned} \tag{32}$$

Differentiating Eq. (32) with respect to each element of $\Delta\boldsymbol{K}$ and equating the results to zero gives the equations that $\Delta\boldsymbol{K}$ must satisfy when ψ is minimal as

$$\begin{aligned}\boldsymbol{K} = \boldsymbol{K}_A &- \sum_{i=1}^{n_T}\sum_{j=1}^{n_T}\nu_{ij}(\boldsymbol{H}_{ij} + \boldsymbol{H}_{ij}^T) \\ &- \sum_{i=1}^{n_A}\sum_{j=1}^{n_T}\lambda_{ij}(\boldsymbol{W}_{ij} + \boldsymbol{W}_{ij}^T)\end{aligned} \tag{33}$$

Where

$$\boldsymbol{H}_{ij} = <\boldsymbol{\phi}_i\boldsymbol{\phi}_j^T> \tag{34}$$

and matrix $\boldsymbol{w}$ is expressed by using the j-th mode as

$$\boldsymbol{W}_{ij} = <\begin{bmatrix} 0 \\ \phi_{j1}\ \phi_{j2}\ \cdots\ \cdots\ \phi_{jn} \\ 0 \end{bmatrix}> \quad \text{i-th row} \tag{35}$$

Taking into account the constraints, one obtains

$$\left.\begin{aligned}&\sum_{i=1}^{n_A}\sum_{j=1}^{n_T}\lambda_{ij}\boldsymbol{T}_{ij} + \sum_{i=1}^{n_T}\sum_{j=1}^{n_T}\nu_{ij}\boldsymbol{B}_{ij} \\ &= \boldsymbol{\phi}^T\boldsymbol{K}_A\boldsymbol{\phi} - \boldsymbol{\phi}^T\boldsymbol{M}\boldsymbol{\phi}\boldsymbol{\Omega}^2 \\ &\sum_{i=1}^{n_A}\sum_{j=1}^{n_T}\lambda_{ij}(\boldsymbol{W}_{ij} + \boldsymbol{W}_{ij}^T)\boldsymbol{\phi} + \sum_{i=1}^{n_T}\sum_{j=1}^{n_T}\nu_{ij}(\boldsymbol{H}_{ij} + \boldsymbol{H}_{ij}^T)\boldsymbol{\phi} \\ &= \boldsymbol{K}_A\boldsymbol{\phi} - \boldsymbol{M}\boldsymbol{\phi}\boldsymbol{\Omega}^2\end{aligned}\right\} \tag{36}$$

where

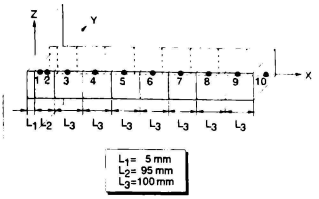


Fig. 1 Finite-element model of the cantilever beam

Table 1 Parameters of the cantilever beam

Parameters		Test	Analysis
Length		800 mm	800 mm
Cross section area		30 mm^2	30 mm^2
Second moment of area	Y	250 mm^4	250 mm^4
	Z	23 mm^4	23 mm^4
Density	E1*1	2.3 X 10^{-6} kg/mm^3	1.4 X 10^{-6} kg/mm^3
	E2*2	2.3 X 10^{-6} kg/mm^3	2.1 X 10^{-6} kg/mm^3
Young's modulus		68600 N/mm^2	75460 N/mm^2
Poisson ratio		0.3	0.3

*1 Elements containing nodes 1 to 6
*2 Elements containing nodes 6 to 10

$$\boldsymbol{T}_{ij} = \boldsymbol{\phi}^T(\boldsymbol{W}_{ij} + \boldsymbol{W}_{ij}^T)\boldsymbol{\phi} \tag{37}$$

$$\boldsymbol{B}_{ij} = \boldsymbol{\phi}^T(\boldsymbol{H}_{ij} + \boldsymbol{H}_{ij}^T)\boldsymbol{\phi} \tag{38}$$

Substitution of the Lagrange multipliers obtained from Eq. (36) into Eq. (33) gives the identified stiffness matrix.

Numerical Example

It is shown in the former chapter that mass matrix can be unique by considering mass property constraints. In this chapter, a numerical example is given to demonstrate that the proposed method is effective for mass property predictions.

Model

The method was applied to a cantilever beam. Figure 1 shows the finite-element model of the cantilever beam. The model has 9 elements and 10 nodes. The beam is made of aluminum. Table 1 shows the parameters of the beam used in the finite element analysis. Half of the outer part of the beam has a density which is different from the density of the inner part. Different input data for the finite-element analysis were taken into account to construct the simulated test and analysis models. We used the lower four frequencies and their

modes as test results for the system identification.

Mass properties

Mass matrix was identified for two cases in which a different constraint was considered: one was the mass property and the other was the mode orthogonality. The mass properties are shown in Table 2. It can be seen that the predicted beam weight agrees well with the measured weight for mass property constraints. However, we could not obtain the predicted weight coincident with the measured ones for the mode orthogonality conditions. Moreover the constraint with mode orthogonality generates physically unreasonable results. The weight has different values in different directions though it should be the same value for all directions.

Table 2 also shows that the weight in the X direction has no change from its initial weight. The cause is considered as follows. For the mode orthogonality conditions, it can be seen from Eq. (7) that the elements of $\boldsymbol{M}$ are corrected from only mode shapes. Because deformation in the X direction is very small for the cantilever beam, the elements in the X direction of the mass matrix cannot be corrected. In fact, the lower four modes used in this case are bending (one bending mode in the Y direction and three bending modes in the Z direction). As the beam deforms in either the Y or Z direction, the weights in those directions are corrected. Three of the four modes considered are modes in the Z direction. Therefore, the weight in the Z direction approaches the measured weight. This means the weight may agree with the measured weight when we consider all the modes.

The center of gravity (CG) in the X direction has two different values because weight and angular momentum are not identical for all directions. The CG value indicated in Table 2 is the larger one obtained from the angular momentum about Z axis (the smaller CG value is 444 mm). In both cases, the CG moves away from the measured value. This indicates that the mass at a point closer to the tip of the beam increases. Namely, the larger the deflection is, the better the mass, at that point, is corrected.

From the above results, we can say that the identified mass matrix can give mass properties mating with the measured ones by the introduction of the mass property constraints.

Natural frequency

Natural frequencies, calculated by using the identified mass and stiffness matrices, are shown with the test results in Fig. 2. The stiffness matrix was identified by using the measured four frequencies and their modes. The updated lower four frequencies are the same as the measured frequencies for both constraints. The natural frequencies are found by

$$\omega^2 = \frac{\boldsymbol{\phi}^T \boldsymbol{K} \boldsymbol{\phi}}{\boldsymbol{\phi}^T \boldsymbol{M} \boldsymbol{\phi}} \tag{39}$$

Therefore, the lower four frequencies have the same values with the test results even though we use any normalized mode. However, for the mass

Table 2 Mass properties

Item		Test	Update[1]	Update[2]	Original[3]
Weight (g)	X	55.2	55.2	41.4	41.4
	Y	55.2	55.2	44.0	41.4
	Z	55.2	55.2	52.3	41.4
Center of gravity (mm)	X	400	400	453	440
	Y	0	0	0	0
	Z	0	0	0	0
Moment of inertia ($X10^3$ kgmm^2)	X	0	0	0	0
	Y	11.8	11.8	12.8	10.2
	Z	11.8	11.8	11.3	10.2

constraints: [1]mass properties
[2]mode orthogonality
[3]result using initial (unupdated) analytical model

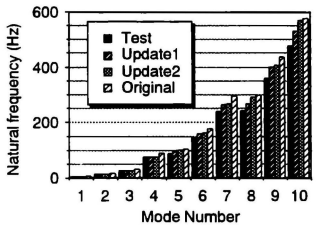


Fig. 2 Natural frequency of the cantilever beam

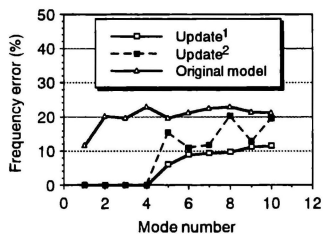


Fig. 3 Frequency error

property constraints, natural frequencies out of the frequency range of interest are closer to the test results than those for the mode orthogonality condition.

Figure 3 shows the frequency error expressed by

$$\text{Error}(\%) = \frac{f - f_{Test}}{f_{Test}} \times 100 \tag{40}$$

The error is zero for the frequencies of interest. For the mass property constraints, the maximum and minimum errors are 11% and 6% respectively. On the other hand, the maximum error is 20% and the minimum error is 11% for the mode orthogonality conditions. The proposed method provides smaller frequency errors concerning higher frequencies. This method appears to have potential as an important tool for identifying mass and stiffness matrices which are closer to the true ones.

Conclusions

This paper proposes a method to obtain a unique solution for system identification. Mass and stiffness matrices are identified by minimizing the Euclidean norm of these matrices subject to some constraints. Mass properties are introduced into the constraints. The following conclusions are obtained:

(1) The identified mass matrix is not unique when mode shapes are used in constraints. This mass matrix does not generate mass properties obtained by the measurements. Moreover the identified mass matrix provides physically reasonable results. These problems can be solved by introducing the mass properties of the structures into constraints.

(2) Stiffness matrix, identified under constraint such as the dynamic equation and the mode orthogonality conditions, is always unique even if any normalized mode is used.

(3) The identified mass and stiffness matrices can provide a highly accurate finite-element model mating with both the modal test and the mass property test results. The proposed method is effective for system identification.

References

[1] Alvin, K. F., "Finite Element Model Update via Bayesian Estimation and Minimization of Dynamic Residuals," AIAA Journal, Vol. 35, May 1997, pp. 879-886.

[2] Kuo, C. P., and Wada, B. K., "Nonlinear Sensitivity Coefficients and Corrections in System Identification," AIAA Journal, Vol. 25, November 1987, pp. 1463-1468.

[3] Baruch, M., "Optimal Correction of Mass and Stiffness Matrices Using Measured Data," AIAA Journal, Vol. 20, November 1982, pp. 1623-1626.

[4] Berman, A., and Nagy, E. J., "Improvement of a Large Analytical Model Using Test Data," AIAA Journal, Vol. 21, August 1983, pp. 1168-1173.

[5] Zhang, O., Zerva, A., and Zhang, D., "Stiffness Matrix Adjustment Using Incomplete Measured Modes," AIAA Journal, Vol. 35, May 1997, pp. 917-919.

[6] Misawa, M., "Mass Matrix Identification Using Test Data," Proceeding of the 26th Space Structures Conference, 1984, pp. 184-187.

[7] Kabe, A. M., "Stiffness Matrix Adjustment Using Mode Data," AIAA Journal, Vol. 23, September 1985, pp. 1431-1436.

[8] Baruch, M., "Modal Data are Insufficient for Identification of Both Mass and Stiffness Matrices," AIAA Journal, Vol. 35, November 1997, pp. 1797-1798.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37238

AIAA-2000-4179

DYNAMIC MODELING AND STABILITY ANALYSIS OF Z-SAT

A. Alasty*, K. Daneshjou†, S. Rezaei‡

Sharif U. of Tech. (SUT), Dep. Of Mechanical Eng. and
Advanced Electronic Research Center (AERC)
Tehran , IRAN

Abstract

In this paper, process of dynamic modeling and stability analysis of Z-SAT is reviewed. Orbit of the satellite is circle, near polar, and about 700 km. above the surface of the earth. A combination of gravity-gradient and magnetorquing stabilization methods is used to attitude control of the satellite. Using a detailed analysis on Keplerian and main perturbation forces, orbital dynamics of the satellite is developed. Proper configuration is designed to take advantages of gravity-gradient torque. Deployable mechanism of control boom, and optimum attitude to deploy it are determined. Several algorithms are depicted to implement different attitude maneuvers, such as changing the rate and direction of rotation, and deliberation. Results of simulations show effectiveness of the stabilization method.

Nomenclature

β , Angle between the symmetry axis and earth magnetic vector
I_a , Axial mass moment of inertia
I_t , Transverse mass moment of inertia
k_0 , Gain of magnetorquer
M_z , Moment of axial magnetorquer
p , Pitch deviation
r , Roll deviation
T_f , Duty time of magnetorquer
ω_c , Orbital angular velocity
ω_s , Axial angular velocity

Introduction

Generally entrance of countries into the field of design and construction of satellites has been begun with design of micro ones (less than 100 kg. satellites). Results of dynamic and stability analysis of such satellites have been published in different proceeding of conferences[1,2].

The gravity-gradient stabilization method is the most conventional selection to stabilizing attitude of near-earth micro satellites. This passive method is usually combined with an active method to optimize speed and accuracy of the control system.

Z-SAT is from the category of micro satellites and is to move in a near circular orbit of 700 km. altitude. Due to its earth-resources imaging mission, passive gravity-gradient method along with active magnetorquing method is advised for this axisymmetric spinning satellite.

The idea of using magnetorquers the first time was depicted by Mr. Psiaki[3]. Magnetorquers are indeed magnetic coils. Crossing of electrical current into them builds magnetic field around them. Interaction of this field and magnetic field of earth makes the control torque. So, controlling the direction, duration, and time of the current would result in appropriate direction and magnitude of the torque.

Uo-SAT2 was of the first satellite that this active method has been used on that. Mr. Hodgart described its control algorithms in 1998[4]. This paper uses some of these algorithms with some modifications.

In this paper, first, Keplerian parameters of the orbit and results of effects of perturbations (such as earth nonhomogeneity, atmospheric drag, and solar radiation) are determined. Then, performance of the satellite is divided in three sections. These three sections are as follows: putting the satellite in a proper attitude before deployment of the control boom, deployment of the boom, and putting the satellite on earth radial direction.

Configuration of Z-SAT

Schematic configuration of the satellite has been drawn in figure (1). The satellite is to move in a sunsynchronous orbit. Its mass is some 60 kg. and mass moments of inertia (before deployment of the boom) with respect to symmetry and transverse axes are some 1 and 2 kg. m^2 , respectively. At the end of telescoping boom a mass of 2.6 kg. is located which increases transverse moment of inertia to some 134 $kg.m^2$.

Orbital Mechanics

A subtle orbital mechanics analysis has been done on this satellite[5]. Eccentricity is assumed 0.01, in the reason of disability of launcher in providing a mere circle. Regarding to sunsynchronous restriction, inclination angle must be 98 degree. Having the satellite on top of Tehran on the first orbital motion, right ascension of the ascending node must be near 228.5 degree.

More than earth gravity, which varies, right ascension angle, effects of atmospheric drag and solar radiation have been considered, too. Effect of gravity of celestial masses (other than earth) on Z-SAT is so small that it has been neglected. Because of the low altitude, atmospheric drag is the dominant

* Assistant Professor, Dep. of Mech. Eng., SHARIF University of Technology.
† Head of Structural Department of AERC, also Assistant Professor, Dep. of Mech. Eng., IRAN University of Science and Technology.
‡ Senior Researcher of AERC, also MSc. of Mech. Eng., Dep. of Mech. Eng., SHARIF University of Technology.
§ Copyright © 2000 The American Institute of Aeronautics and Astronautics Inc. All rights reserved.

perturbation resource. Variation of Keplerian parameters because of the perturbations has been shown in figure (2) and (3). These figures show that altitude of the orbit is decreased 2 km. at the end of mission. This analysis revealed that assumption of ideal orbit instead of actual one in the future attitude simulations would not be an origin of error.

Gravity-Gradient Stabilization

Augmenting stabilization effect of the earth gravity field, a long control boom has been used for Z-SAT. Its deployment mechanism has been selected telescoping type to make that able to endure external loads during the transient deployment time. This mechanism does not increase mass and volume of Z-SAT too high. Deployment phase of the boom has many problems. Stability analysis shows that Z-SAT must be at proper attitude before deployment. The right initial attitudes would result in preventing from tumbling. Deployment is carried on constant rate and at 15 min.. In order to determine attitude behavior of Z-SAT, two angles are defined: Pitch and Roll. Pitch deviation or angle is the angle between projection of symmetry axis on orbital plane and earth radial direction, and roll deviation is deviation of the axis from the plane (figure (4)).

So, using the Euler equations for the rigid satellite provides attitude equations as bellow. In this section, gravity torque is the only external torque.

$$I_t\ddot{p}Cosr = -I_a\dot{r}\omega_a + 2I_t(\omega_c + \dot{p})\dot{r}Sinr - \dot{I}_t(\omega_c + \dot{p})Cosr - 3(I_t - I_a)\omega_c^2 SinpCosrCosp$$

$$I_t\ddot{r} = I_a\omega_a(\omega_c + \dot{p})Cosr - I_t(\omega_c + \dot{p})^2 SinrCosr - \dot{r}\dot{I}_t - 3(I_t - I_a)\omega_c^2 Cos^2 pSinrCosr \quad (1)$$

Axial rotation of Z-SAT helps sensibly in suppression of tumbling. Figure (5) shows results of simulation. Optimum period of the axial rotation is five minutes and initial pitch and roll angles are 60 and 30 degree, respectively. These values have provided the best response.

Control booms are usually damaged by thermal shocks. Despite of long length and narrow diameter of the boom, no severe thermal problem was found on Z-SAT, because the axial rotation removes thermal gradient across the boom.

As figure (6) shows, variation of boom mean temperature is not so high in the orbit[6]. Variation of temperature across the boom has a better situation.

Magnetorquing

Three orthogonal coils are used in Z-SAT. Before deployment of the boom, they are used to orienting the symmetry axis on proper direction. After deployment, they are used to suppress deviation angles and putting the axis on the earth radial direction.

To remove the transverse rotation that occurs after separation of the satellite from launcher, the axial magnetorquer (which its magnetic moment lies on the axial direction) is fired. The algorithm of determining direction of the current is as below,

$$Sgn(M_z)=Sgn(dB_z/dt) \quad (2)$$

Where, "Sgn" function determines sign of parameter and subscript "z" refers to component of the vector along axial direction of the satellite.

Figure (7) shows variation of angular velocities ω_x, ω_y with respect to each other. Initial values of them are 0.0105 rad/s and time of analysis is 3000 sec..

Having removed the transverse rotation, satellite is directed on magnetic vector. The following algorithm does this task,

$$Sgn(M_z)=-1 \text{ ; If } [(B_z>-B_t \text{ \& } Sgn(dB_z/dt)>0) \text{ Or } (B_z<-B_t \text{ \& } Sgn(dB_t/dt)>0)]$$
$$Sgn(M_z)=0 \text{ ; Otherwise} \quad (3)$$

In which, subscript "t" refers to transverse component of magnetic field. Then, using two transverse magnetorquers, the pre-deployment attitude (pitch=60 and roll=30 degree) are provided, without changing rate of the axial rotation.

After deployment of the control boom, when the axial magnetorquer is used, two algorithms are implements. The first one is fruitful for the case of high deviation angles and the other for low ones. The first algorithm is as follows,

$$Sgn(M_z)=-Sgn(dB_z/dt) \text{ ; If } |dB_z/dt|>(dB_z/dt)_T$$
$$Sgn(M_z)=0 \text{ ; Otherwise} \quad (4)$$

Where, subscript "T" refers to threshold value of the related parameter. Selection of duty time of the magnetorquer and threshold variation rate of B_z is the most important considerations of this algorithm. Optimum values of them are 3 min. and 55.9 nT/sec., respectively. In this analysis, components of magnetic vector have been derived from I.G.R.F model. Figure (8) shows results of the simulation. Initial values of pitch and roll angles are 60 and 30 degree, respectively. Results have been drawn in two different regions of time for clarity. The mentioned algorithm deoreases initial angles very much in just two duty days of the satellite.

To diminish the lower angles (provided at the end of implementation of the first algorithm), rate of variation of the β angle is compared. Thus, the second algorithm is as follows,

$$Sgn(M_z)=Sgn(d\beta/dt) \text{ ; If } |d\beta/dt|>(d\beta/dt)_T$$
$$Sgn(M_z)=0 \text{ ; Otherwise}$$
$$T_f=k_0(|d\beta/dt|-(d\beta/dt)_T \quad (5)$$

The threshold value of $d\beta/dt$ is found from I.G.R.F model using the bellow numerical formula. This assumes the ideal attitude case (having no deviation),

$$d\beta/dt=(B_{zc}-B_{zp})/(B_{tc}+B_{tp}) \quad (6)$$

Where, subscripts "c" and "t' refer to current and previous values in numerical simulation of magnetic field. Figure (9) shows variation of the threshold values on the orbit of Z-SAT. Results of the simulation based on the current algorithm have been shown in figure (10). Amplitudes of the pitch and roll angles are reduces down to one and two degree, respectively.

In this algorithm, duty time of the magnetorquers is adjusted proportion to deviation. The optimum value of this gain is 1500 sec.^{2}. Initial values of this algorithm are final values of the previous algorithm.

Conclusion

In this paper, process of stability analysis of Z-SAT was described. Its important characteristics, such as rate of the axial rotation, time, and proper pre-deployment attitude were determined. Results show that combination of passive gravity-gradient and active magnetorquing methods could result in acceptable performance for this low-cost micro satellite. Magnetorquers lead the satellite from completely unideal case to ideal case of lying the symmetry axis on earth radial direction. Accuracy of these stabilizing methods is about one and two degree for pitch and roll angles, respectively.

Implementation of the theoretical considerations mentioned in this paper, on Z-SAT needs some practical heed following the current research. First, determining dynamics of sensor of magnetic vector (magnetometer) and its calibration method. Second, using data of magnetometer or every other sensor to estimation of attitude and including that in algorithms of firing of magnetorquers.

Acknowledgment

This research has been sponsored by Advanced Electronic Research Center of Iran.

References

[1] Whisnant, J.M., Waszkiewicz, P.R., and Piscane, V.L., "Attitude Performance of the GEO-II Gravity-Gradient Spacecraft," *Journal of Spacecraft and Rockets*, Vol. 6, No. 12, pp. 1379-1384, 1969.

[2] Markley, F.L., Bauer, F.H., Deily, J.J., and Femiano, M.D., "Attitude Control System Conceptual Design for Geostationary Operational Environmental Satellite Spacecraft Series," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 2, pp. 247-255, 1995.

[3] Martel, F., Pal, P., and Psiaki, M., "Active Control System for Gravity-Gradient Stabilization Spacecraft," *Second Annual AIAA/USU Conference on Small Satellites*, Washington, DC: AIAA, Unnumbered, 1988.

[4] Hodgart, M.S., "Gravity-Gradient and Magnetorquing Attitude Control for Low-Cost Low-Earth Orbit Satellites: the UOSAT Experience," *Proceeding of the AIAA/AAS Astrondynamics Conference*, Washington, DC, pp. 421-430, 1988.

[5] Rezaei, S., "Dynamic Modeling and Stability Analysis of a Sample Satellite with Considerations of Earth Orbit Effects," *Sharif University of Technology*, Department of Mechanical Engineering, M.Sc. Thesis, April 2000, (In Persian).

[6] Hosseini, A., "Design of Thermal Control System of a Sample Satellite Using Lumped Parameters Method," *Advanced Electronic Research Center*, Technical Report, No. 110, Sep. 1999, (In Persian).

Figures

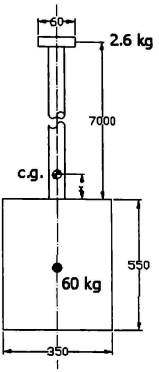


Figure 1, Schematic configuration of Z-SAT

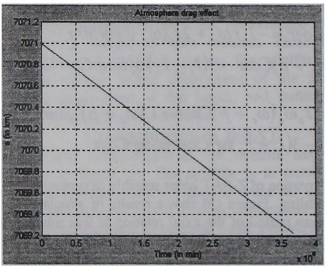


Figure 2, Variation of Keplerian parameters because of atmospheric drag

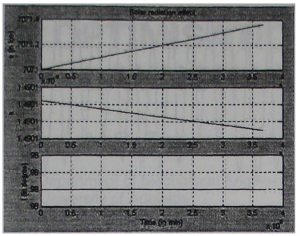


Figure 3, Variation of Keplerian parameters because of solar radiation

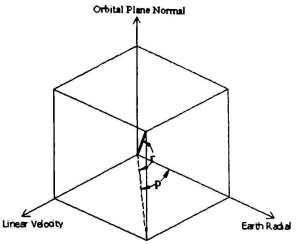


Figure 4, Deviation angles of Pitch and Roll

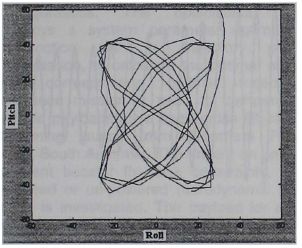


Figure 5, Variation of pitch angle versus roll angle during deployment time

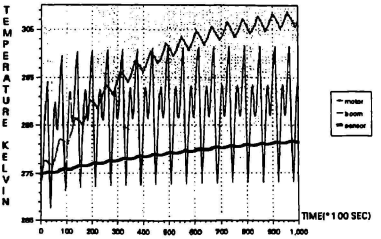


Figure 6, Variation of boom temperature in the orbit

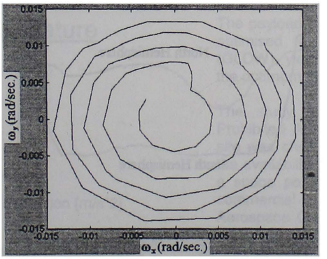
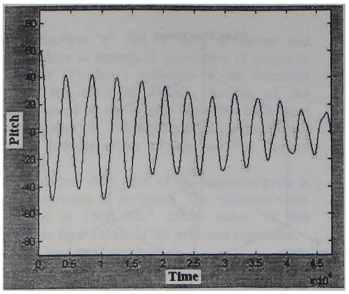
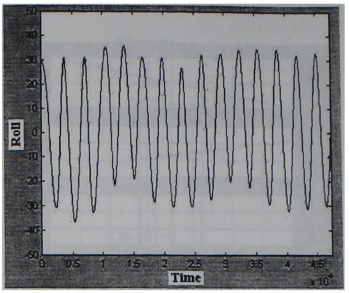
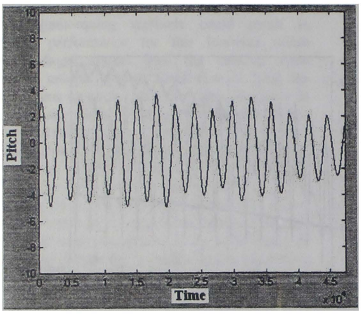


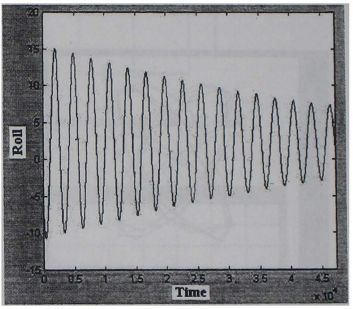
Figure 7, Variation of ω_y versus ω_x





Start Time=0 ; Finish Time=790 min.





Start Time=3160 min. ; Finish Time=3950 min.

Figure 8, Variation of pitch and roll angles versus time (based on the first algorithm)

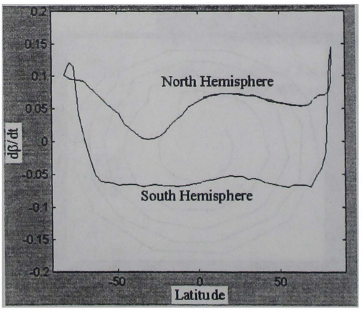


Figure 9, Rate of variation of the β angle versus latitude (in ideal case)

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37239 AIAA-2000-4180

OPTIMISATION OF A REUSABLE LAUNCH VEHICLE CONCEPT

M.R. Tetlow*, U.M. Schöttle+, G.M. Schneider*

*Department of Mechanical Engineering
University of Adelaide
Adelaide, Australia

*Space Systems Institute
Stuttgart University
Stuttgart, Germany

Abstract

The aim of the present study is to investigate alternative reusable launch vehicle concepts and to provide a comparison of two different vehicle configurations. The optimisation software employs a system parameter optimisation in conjunction with a trajectory parameter optimisation, to calculate the optimal mass split and corresponding optimal trajectory. A reference mission is used to compare launch vehicle payloads for a fixed gross lift off weight, assuming launch from Woomera Prohibited Area, South Australia. The impact on payload for different booster fly back constraints, such as powered or un-powered (aerodynamic glide) fly back, is investigated. The payload for a vehicle using a powered fly back strategy is found to be approximately 50% higher than that obtained by a vehicle employing an aerodynamic glide return flight. The optimum staging velocity for the powered return concept is found to be approximately 3000 m/s. The optimum for the glide back concept is found to be at the highest staging velocity possible while still being able to perform a safe return flight.

Nomenclature

m = Mass [kg]
h = Altitude [m]
v = Velocity [m/s]
L = Lift force [N]
D = Drag force [N]
I_{sp} = Specific impulse [s]
g_0 = Gravitational acceleration [m/s^2]
s = Distance [m]
Δv = Velocity change [m/s]

Introduction

Recent years have seen a significant increase in the number of launch vehicles available for commercial payloads, resulting in a highly competitive launch vehicle market. With the exception of the Space Shuttle, all current launch vehicles, such as Ariane, Delta, H-2, Long March, etc. are expendable[1] and all current launch vehicles are expensive to operate. This fact suggests that there is a need for a fully reusable launch vehicle that can provide a cheap reliable launch service.

To contribute to the examination of future launch systems, the impact of mission requirements on launch vehicle design was examined. Two concepts of winged reusable launch vehicles, employing vertical take-off and horizontal landing, were compared using different fly back strategies for the booster stage. The first concept used air breathing engines for a powered return flight, while the second employed an un-powered aerodynamic glide to return to the launch site. To enable un-powered fly back of the booster, staging had to be performed sufficiently early to be close enough to the launch site for glide back. The payload capability of the launch vehicle was optimised for a fixed Gross Lift Off Weight (GLOW), yielding optimum stage mass split and the optimum trajectories for each concept.

The study assumed launch from Woomera Prohibited Area in South Australia. This launch site was chosen due to its remote location and the availability of the necessary infrastructure for a space port. Recent interest in Woomera by commercial launch companies, such as Kistler Aerospace Corporation, also justified the use of this launch site.

Copyright © 2000 by the American Institute of Aeronautics and Astronautics Inc. All rights reserved

Problem Description

An assessment of the commercial payloads expected for the next 10 years, carried out by the FAA[2], predicts that 76% of satellites launched into Low Earth Orbit (LEO) will have a mass less than 800kg. The Geosynchronous Earth Orbit (GEO) market will be dominated by two categories. It is expected that 36% of payloads will be between 1815 – 4082kg and 42% between 4082 – 5443kg, with the FAA prediction also indicating a growth in the number of satellites above 5443kg in the GEO category.

There are a number of launch vehicles that either already service the commercial payload market or are expected to in the near future. Table 1 shows a performance comparison of these vehicles.

Description		Payload		Cost
Name	GLOW [ton]	LEO [kg]	GTO [kg]	Mil $US
Ariane 5	737	16000	6800	150-180
Delta IV	733	19200	10843	140-170
BA-2	1170	17000	5800	[-]
Shuttle	2040	16050	[-]	300
A	406	8400	4100	165-170

Table 1 Launch vehicle comparison[1]

The high cost of these launch vehicles is due to their expendability or their requirement for extensive refurbishment before launch. This high cost has prompted much research into the development of fully reusable launch vehicles e.g. NASA's X-33 test vehicle and the FESTIP[3] study carried out in Europe. An indication of the cost reduction of reusability is displayed by the Kistler K1 reusable vehicle. According to Isakowitz[1], it will cost Mil $US 17 for a 4.5 ton payload to LEO, a significant reduction compared to the values in Table 1.

Reference Mission

Considering the FAA study, it would appear that a reusable launch vehicle needs to be developed to address the GEO payload market. The high LEO capability of such a launcher could also be used to service the International Space Station (ISS). It was decided to use an expendable third stage due to the high mass of the second stage, which would require large volumes of propellant to achieve Geostationary Transfer Orbit (GTO). Clearly, this would significantly reduce the GTO payload capability. The two stage system was thus optimised to achieve LEO.

The reference mission chosen was a 466km circular mission inclined at an angle of 31.5°. The ascent trajectory was optimised for perigee insertion into a 94km x 466km orbit, with an allowance made for the fuel required to circularise to a 466km circular orbit. A similar mission was used by Freeman et al. (1995)[4] for a conceptual design of a manned launch vehicle, providing a base for comparison.

The payload capability for GTO was based on a 466km x 36000km altitude orbit. In this case the payload for the 466km circular orbit was used as the initial mass of the expendable third stage. The required fuel and structure weight for the third stage was calculated using a structural coefficient of 15% and an I_{sp} value of 324s[1] for the propulsion system. It was also assumed that the payload would perform the inclination change from the 31.5° GTO to a 0° GEO.

The booster fly back for concept 1 assumed an aerodynamic turning glide flight until reaching sub-sonic cruise conditions. The remainder of the return flight was performed at cruise conditions powered by air breathing engines. Concept 2 employed only aerodynamic glide flight to achieve fly back. In both cases considered in this study, the vehicle was assumed to land at the launch site at a velocity of 120 m/s.

Concept vehicle

The unmanned concept vehicles discussed in this paper had a gross lift off weight of 1400 tons. This value was chosen to enable the vehicles to launch approximately 20 tons into LEO, thereby allowing them to address the GEO market requirements discussed in Middleton et al.[2]

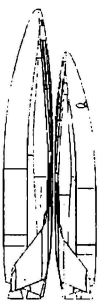


Figure 1 Vehicle concept

They consisted of two winged stages which operated in parallel during launch and initial

ascent (Figure 1). A cross-feed fuel system was employed to allow the orbiter to use propellants stored in the booster, during mated flight. A cross-feed system was chosen as it was found to be optimum for a 2 stage vehicle[3]. The mated vehicle employed 11 liquid oxygen (LOX)/ kerosene rocket engines for the ascent flight, providing a thrust to weight ratio of approximately 1.2. The number of rocket engines on each stage depended upon the stage size. The powered return vehicle concept also employed 2 or 3 kerosene air breathing engines for fly back, depending on the size of the booster stage.

Although a winged orbiter was not necessary, it would provide greater flexibility for the re-entry mission as opposed to a ballistic re-entry vehicle. A winged re-entry vehicle would thus be advisable for a launch system that required a short turn around time. The orbiter conceived had the capability to land with its full launch payload. This was done to enable the vehicle to return a payload from orbit and also to avoid the payload having to be dumped in the event of an aborted launch.

Simulation and Optimisation Techniques

The simulation technique used in the present study employed a point mass, rigid body dynamic model. The Earth was modelled as a rotating sphere with a Newtonian approximation of its gravitational field. A static atmosphere model, similar to the US standard atmosphere, was employed with no wind model. The 3 Degree Of Freedom (DOF) dynamics equations were integrated using a Fourth Order Runge Kutta numerical integration technique.

Mass modelling

Statistical mass modelling was employed to calculate the vehicle subsystem masses. A least square curve fitting technique was used to linearise statistical mass data, which could be chosen by the launch vehicle designer. This choice allowed the designer to choose data which was perceived to have a similar technology level to the vehicle being considered. The mass model could also be scaled up or down, by a factor, to represent a change in technology. Subsystem masses m_i were calculated using the statistical data in the form shown in Equation 1:

$$m = \sum_{i=1}^{n} m_i = \sum_{i=1}^{n} A_i . x_i^{B_i} \qquad (1)$$

where A_i and B_i are the coefficients that define the technology, with respect to property x, of the subsystem.

The mass model for the present study was based on the FESTIP FSSC 16[3] concept vehicle. Vehicle sizing was calculated from the amount of propellant required to be stored in the vehicle's tanks, including a 4% reserve. Assumptions had to be made regarding the mass of the non-structural kerosene tanks, due to the lack of available data. The tanks were assumed to have the same mass fraction as that of the oxygen tanks. Due to the fact that kerosene can be stored at lower pressures than oxygen and that kerosene tanks can be manufactured from composite materials[5], this assumption was considered to be conservative but satisfactory.

The mass of the fuel required for fly back of the powered return concept, was calculated using the relationship shown by Equation 2:

$$\Delta s = \frac{L}{D}\left(\frac{v_1^2 - v_2^2}{2g} + h_1 - h_2 + I_{sp} g_0 \ln \frac{m_1}{m_2} \right) \qquad (2)$$

The indices 1 and 2 represent conditions at the end of the aerodynamic glide trajectory and at the end of cruise flight respectively. Thus, the propellant consumed during fly back was the difference between m_1 and m_2.

Aerodynamic modelling

The aerodynamics of the vehicle were modelled as a table of lift and drag coefficients for given Mach numbers and angles of attack. These values were then interpolated by the program, using a spline function, to produce a continuous function that described the aerodynamics of the vehicle.

The aerodynamic data used for the following model was taken from the aerodynamic model of the Ariane-X Concept vehicle[6] and then scaled to represent the aerodynamics of the FSSC-1 concept vehicle in the hypersonic region[3]. This scaling resulted in a somewhat optimistic maximum lift to drag ratio of 7 at a Mach number of 0.9.

Propulsion systems

The propulsion systems chosen for this study included both rocket and air breathing systems, depending on the configuration being considered and the flight regime.

The core rocket engines used for the ascent were the Aerojet AJ26 series engines. These

engines are modified Russian NK33 and NK43 engines, as used on the Russian Lunar program. The modifications made by Aerojet include updated electronics, igniters, electronic controller, actuated control valves and ±6° gimbaling[7].

The rocket engines used on the booster stages were the Aerojet AJ26-59 engines, which have an I_{sp} value of 331.3s in vacuum and 297.2s at sea level. They weigh 1459kg and produce 1.68MN of thrust in vacuum and 1.51MN at sea level. These engines are suited to low altitude flight and have an expansion ratio of 27. The Orbiter was equipped with AJ26-60 engines, which have an I_{sp} value of 345.3s in vacuum. They produce 1.75MN of thrust in vacuum and weigh 1505kg. These engines are suited to high altitude flight and have an expansion ratio of 80[7]. Due to the fact that the mass of the AJ26 engines is known[7], it was not determined using the statistical mass analysis approach, but was simply calculated from the number of engines on the stage.

The air breathing propulsion system used for fly back was based on the performance of the M88-35 turbo jet engine. The mass for each engine was estimated to be 750 kg[3]. Without after burners, this engine can produce 60kN of thrust with a fuel consumption of 80kg/kN.h.

The air breathing engines were not used during the launch phase due to operational constraints. From an optimisation point of view, air breather propulsion during ascent increased the payload, however, the added complexity of advanced intake nozzles made it an unrealistic proposal. For this reason, the air breathing engines were only activated during sub-sonic cruise conditions in the fly back phase.

Previous concept vehicles such as Sänger[6], used their air breathing propulsion system during ascent, descent turn flight[6] as well as cruise flight, thereby suggesting that an engine operating in the higher Mach numbers was optimal. It was however discovered that the propulsion system should be throttled to produce only a fraction of its possible thrust during the supersonic descent phase[6]. Further, Sänger used the air breathing engines for ascent so already had high speed engines for use during descent. From the results of this study, it was decided that high speed air breathing engines were not required, therefore lighter sub-sonic air breathing engines were employed during cruise conditions.

Optimisation technique

The optimisation software used for the present study was developed at the Space Systems Institute in Stuttgart Germany and is described in Rahn et al. (1999)[6]. The task was formulated as a non-linear programming problem that was solved by a sequential quadratic optimisation algorithm employing central difference gradient calculations. The chosen approach combined both optimisation and design steps for the development of a launch vehicle. The payload was optimised for given staging conditions which were then varied to obtain a trend towards the optimum staging conditions.

The mission was broken into 3 flight phases which had different parameter sets and were optimised independently to achieve a solution. Phase 1 was the first stage ascent of the mated configuration. It employed 3 parameters to describe the flight profile, consisting of an initial pitch manoeuvre followed by a gravity turn. The 3 parameters were thrust vector angle, duration of the pitching manoeuvre and the launch azimuth. Phase 2 was the booster stage fly back and it was controlled by 16 optimisation parameters, the first 12 being angle of attack control parameters and the remaining 4 bank angle control parameters. The bank angle was required to control the turn manoeuvre before cruise flight (concept 1) or glide phase (concept 2) conditions were reached. Finally, phase 3, which was second stage ascent, was optimised with respect to 5 angle of attack parameters defining thrust vector angles.

Results

Staging Conditions

The staging conditions play a significant role in the determination of an optimal trajectory for a given mission. Hence, it is important to perform staging at optimal conditions to maximize the payload.

A number of optimisation runs were performed on the orbiter stage using different staging flight path angles, velocities and altitudes. This was done to try and determine an initial estimate for the concept vehicle optimisation. A fixed mass of 450 tons was used with a thrust to weight ratio of 1.2. According to Schöttle (1989)[8], the optimal flight path angle is between 7.5° and 20° for staging velocities between 2 km/s and 3.5 km/s. There is a slight shift towards lower flight path angles for an increase in staging velocity and staging altitude.

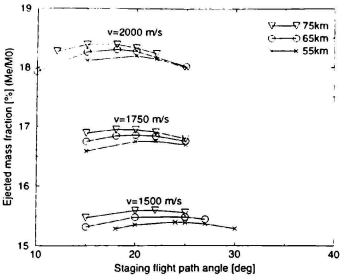


Figure 2 Staging condition comparison

The results displayed in Figure 2 show a similar trend to those seen in Schöttle (1989)[8] which used higher energy cryogenic propellants. The ejected mass fraction is shown to increase with an increase in staging velocity and altitude. The staging flight path angle is shown to have an optimal value around 20° with a shift to lower flight path angles with an increase in staging velocity and altitude.

Concept 1

The first concept considered was a two stage, winged vehicle in parallel configuration employing a powered booster return flight. The booster fly back was achieved using air breathing propulsion.

A powered return flight was proposed to allow high staging velocities. Three different staging conditions were optimised to determine the design point for the configuration.

Mission using powered booster fly back			
Staging velocity [m/s]	2500	3000	3500
Staging flight path angle [deg]	15	10	8
Staging altitude [km]	59,8	62,8	70,0
Booster mass [tons]	1064	1123	1182
Booster propellant [tons]	979	1039	1093
Orbiter mass [tons]	336	277	218
Orbiter propellant [tons]	298	209	184
Fly-back distance [km]	682	841	1064
Maximum load factor [g]	3,5	3,6	3,6
Number of air-breathers	2	2	3
Payload ISS [kg]	17943	21693	20822
Payload GTO [kg]	5793	7005	6723

Table 2 Vehicle comparison

The results from the optimisations, shown in Table 2, indicate that the optimum staging condition for a vehicle of this configuration is at approximately 3000m/s.

Previously it was shown that the higher the staging velocity, the higher the injected mass fraction. However, it was also obvious that the higher the staging velocity, the higher the first stage mass, due to larger propellant tanks. This higher mass required a larger volume of propellant for the fly back manoeuvre. Another factor contributing to the vehicle becoming heavier for higher staging velocities, is the greater number of air breathing engines required to overcome aerodynamic drag during the return flight. For the vehicle staging at 3000m/s, only 2 engines were required for fly back, while the higher staging velocity of 3500m/s required 3 air breathing engines.

Table 2 shows a relatively small difference between the payload for a 3000m/s staging velocity and that for a 3500 m/s staging velocity. This may indicate that the payload is not highly sensitive to staging velocity in the region of the optimum value. In this region therefore, the design of a commercial launch vehicle may be dominated by the operational constraints instead of the payload capability.

The optimal staging velocity obtained in the present study was similar to the value of 3200 m/s obtained by Rahn et al. (1999)[6] for the optimal staging velocity of the Ariane-X concept vehicle. The Ariane-X vehicle also displayed a relatively small range of payload variation around the optimum staging velocity value.

The mass model employed for the present study used the landing mass of the vehicle to calculate the required mass of the wings. It considered a maximum load factor of 3.75g for a pull up manoeuvre before landing. An investigation was performed using the 3000 m/s staging velocity, to determine the effect on the payload of higher wing loads during the aerodynamic turn. This was then used to determine if the added wing mass, required to withstand a higher load turn during fly back, would outweigh the increased payload.

The mission was first optimised to allow a 3.6g turn during flight, yielding a payload of 21693kg. The comparison mission was optimised to allow a 4.6g turn, yielding a payload of 21827kg. Hence, the difference in payload was 134kg. The difference in mass between a wing designed to withstand a 3.75g load during landing and one designed to withstand a 4.6g turn during the

return flight was calculated to be 621kg. Clearly it is thus not optimal for this vehicle to perform high load turns during flight, as the extra mass required by the wing, to support the loading, would outweigh the payload advantage from such a turn.

A result noted for the 2500m/s staging velocity indicated that the vehicle did not have to execute a turn during its descent flight. Only a small difference in fuel consumption was noted for a trajectory that included a turn during its descent and one that descended, in the ascent plane, and then executed a sharp turn at cruise conditions.

Design point vehicle

The design point for this vehicle concept was a staging velocity of 3000 m/s, flight path angle of 10° and altitude of 62.8 km.

The vehicle consisted of a 1123 ton booster stage employing 8 AJ26-59 rocket engines and a 277 ton orbiter stage having 3 AJ26-60 rocket engines. The vehicle operated in parallel during phase 1, employing a cross-feed fuel system to enable the orbiter to use propellants stored in the booster. This minimized the structural mass of the orbiter stage, thereby reducing the mass that had to be accelerated to the final orbit conditions.

This design required a return flight of 841km, during which it consumed 7706kg of fuel. The vehicle required 2 air breathing engines to overcome the drag force during cruise flight.

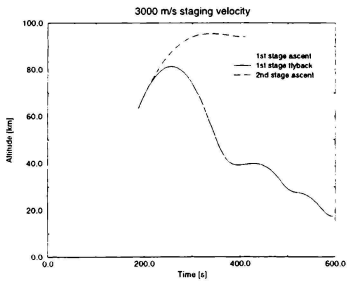


Figure 3 Altitude profile

As figure 3 shows, the vehicle flew in the mated configuration for 186 seconds before separation. The orbiter then switched to its own tanks and continued the ascent to the required orbit of 94km x 466km. The booster performed a coast phase while in the upper atmosphere with an angle of attack of 40° and a bank angle of -60°. The reason for this was to perform as much of a turn and braking manoeuvre as possible, while in the low density atmospheric region. After approximately 330 seconds, the first stage vehicle performed a controlled braking and turning manoeuvre to change the heading towards the launch site and to achieve cruise conditions. Once cruise conditions had been reached, the air breathing engines were started, which propelled the vehicle back to the launch site (not shown in Figures).

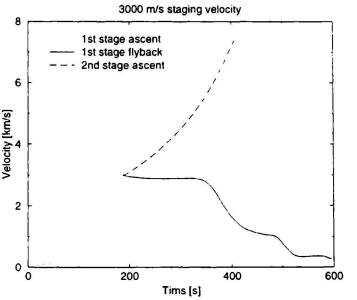


Figure 4 Velocity profile

From Figure 4 it can be seen that the ascent phases, of the first and second stage, show a distinct exponential velocity increase during flight. This is caused by the thrust increasing and vehicle mass decreasing as the vehicle ascends.

Figure 4 also shows the velocity of the booster remained relatively constant, around a value of 3000 m/s for the first 180 seconds after separation. This was due to the low atmospheric density during this flight time, producing low aerodynamic forces on the vehicle.

The ascent of the second stage was throttled during the last 30 seconds of flight to avoid load factors above 3.5g.

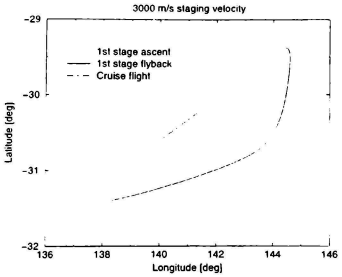


Figure 5 Ground track

Figure 5 shows a significant heading change close to the termination of the optimisation. The reason for this is a high lift force at this altitude, due to the high atmospheric density. It was also a suitable time to execute a turn manoeuvre as the velocity of the vehicle was relatively low thereby resulting in a lower load factor during the turn.

It can be seen from Figure 5 that from staging to approximately 144° longitude, the booster stage followed the ascent plane instead of executing a turn manoeuvre. Although there was a high bank angle and angle of attack, as shown in Figure 7, low atmospheric density of the upper atmosphere produced little turning force.

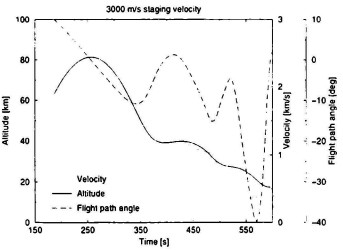


Figure 6 Stage 1 fly back profile

From Figure 6, it can be seen that the booster climbed to an altitude of 81km before beginning its descent. At this high altitude, the lift forces on the vehicle were low, typically in the range of 23KN, thereby making control less effective in this area. This was evident by a relatively low sensitivity to control parameters in this flight regime.

The optimisation was performed with the constraint of achieving a cruise velocity of 270m/s and a flight path angle of 0°, at the termination of the simulation. A final altitude was not specified as an optimisation constraint to allow more freedom of optimisation. Provided the termination was above the specified cruise altitude, the solution was accepted. The vehicle was assumed to descend slowly at cruise conditions until it reached the required cruise altitude of 12500 m. This accounted for the difference between the altitude at which the simulation terminated and the required cruise altitude.

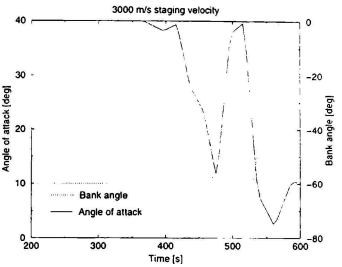


Figure 7 Control parameters for fly back

In Figure 7, it can be seen that the bank angle control parameter started at a value of –60°. This was arbitrarily chosen as the bank angle value before the optimisation parameters became effective. The reason for the –60° bank angle was to perform as much of a heading change as possible during this flight period. Initially, an optimisation parameter was used in this early flight region, but it proved problematic to the optimiser due to low optimisation sensitivity of the parameter. It was thus decided to use a constant value for the angle of attack and bank angle until the aerodynamic forces became significant enough to allow stable optimisation.

From Figure 7, it can be seen that once the optimisation for the bank angle parameters was activated, at 330 seconds, the bank angle jumped to a value of approximately -28°. This indicates that the bank angle assumed before the optimisation process started could have been reduced to a value of around –30°.

The peak in the angle of attack at around 500 seconds was required to create sufficient lift for the vehicle while it was at the maximum bank angle of –60°. This served to produce a large turning force, while at the same time preventing a steep descent.

Concept 2

The following concept used an aerodynamic glide to achieve the booster stage return flight. The vehicle concept was investigated using 3 different staging conditions, the results of which are shown in Table 3.

The staging flight path angle was optimised for the ascent phases only. The optimum flight path angle, with respect to ascent, was then used to determine if a return glide to the launch site was possible. The maximum distance that the vehicle could glide was determined using Equation 2. Clearly, the mass before and after the return flight was the same, thereby eliminating the log term.

As for the powered return flight concept vehicle, the stopping criteria for the optimisation was a velocity of 270 m/s. This velocity was used as the initial velocity in Equation 2. The final velocity was taken to be 120 m/s which was the assumed landing velocity.

Gliding booster fly back mission			
Staging velocity [m/s]	1000	1100	1200
Staging flight path angle [deg]	27	27	27
Staging altitude [km]	24,6	30,62	34,0
Booster mass [tons]	729	770	799
Booster propellant [tons]	668	708	736
Orbiter mass [tons]	671	630	601
Orbiter propellant [tons]	588	549	521
Fly-back distance [km]	69,5	80,9	92,28
Maximum load factor [g]	2,7	2,6	3,4
Payload ISS [kg]	13471	14079	14709
Payload GTO [kg]	4358	4546	4749

Table 3 Vehicle comparison

From Table 3, it is clear that the maximum payload was achieved using the highest staging velocity possible. The limitation of the staging velocity was the ability of the vehicle to glide back to the launch site after staging. This maximum velocity from which a return to launch site glide was possible was found to be approximately 1200 m/s, corresponding to a Mach number of 3.9.

When comparing this value to the value used by Powell et al. (1991)[9], it was found to be considerably higher. Powell et al. (1991)[9], chose a staging Mach number of 3 to allow a glide return flight to the launch site. This made an allowance for a safety factor with respect to the glide distance and was also low enough to limit aerodynamic heating. Low aerodynamic heating eliminated the requirement for a thermal protection system on the booster stage. The present study included a thermal protection system on the booster thereby allowing higher heating loads and possibly accounting for the higher staging Mach number.

Design point vehicle

The design point for this concept was a 1200m/s staging velocity. The flight path angle at this staging condition was found to be optimal at a value of 27°, for the powered ascent flight.

The vehicle design consisted of the same parallel, propellant cross–fed, winged, two stage configuration as used previously. The booster employed 6 AJ26-59 rocket engines and had a lift off mass of 769 tons, while the orbiter had 5 AJ26-60 engines and a lift off mass of 631 tons.

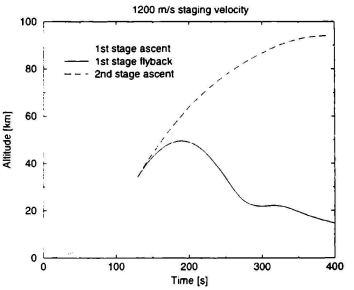


Figure 8 Altitude profile

As Figure 8 and 9 show, the vehicle was accelerated to a velocity of 1200m/s and an altitude of approximately 30km. The vehicles then separated and the orbiter continued along its ascent trajectory, while the booster began its un-powered return flight to the launch site.

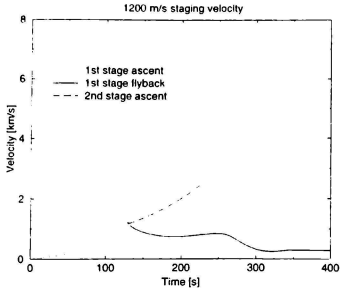


Figure 9 Velocity profile

From Figure 9 it is apparent that the vehicle in the mated configuration produced only a small portion of the total velocity change required for the mission. The remaining velocity change necessary was performed by the orbiter stage alone, thereby making it a significantly larger mass than that for the vehicle with a 3000m/s staging velocity.

As before the engines on the orbiter were throttled for the last 95 seconds of the ascent flight to limit the acceleration of the vehicle to 3.5g.

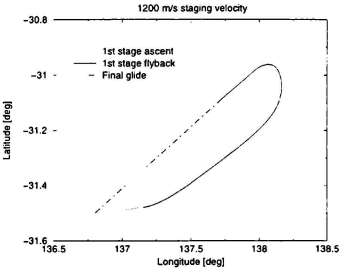


Figure 10 Ground track

From Figure 10 it can be seen that after separation, the booster completed a turn manoeuvre to assume a heading towards the launch site. It then glided a further 8 km before the optimisation was terminated at a velocity of 270 m/s. It was assumed that the vehicle glide phase, after termination of the optimisation, was at an angle of attack that would maximize the return flight distance.

During the first 30 seconds after separation, Figure 10 shows that a significant heading change occurred. This was due to the staging occurring at an altitude with a relatively dense atmosphere. This allowed a turn manoeuvre to be executed before the vehicle coasted into the lower atmospheric density region. In this low density region, a straight flight is observed as was the case for the previous vehicle concept.

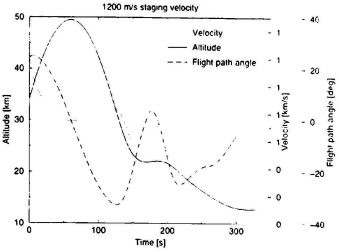


Figure 11 First stage fly back profile

Unlike the velocity profile of Concept 1, Figure 11 shows that the booster stage began decelerating immediately after separation. This was due to the fact that the atmospheric density was relatively high in this region, thereby producing high aerodynamic drag on the vehicle. The aerodynamic drag at the time of staging was approximately 164 MN.

Figure 11 also shows a sharp peak in the flight path angle between 130 seconds and 220 seconds flight time. This skip behaviour was caused by the rate of density increase being faster than the deceleration rate, causing increased lift and the levelling out of the altitude profile. This could be minimized by imposing an upper limit on the flight path angle during the descent stage.

Trajectory optimisation was terminated when the velocity and flight path angle constraints were met close to the desired altitude. Again, final descent and landing was not considered in the simulation.

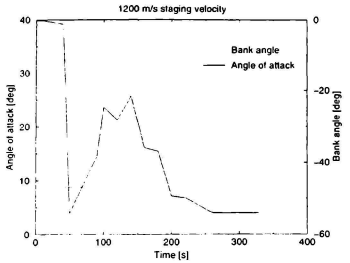


Figure 12 Control parameters for fly back

As with the control model for the powered return flight design, an arbitrary initial bank angle of -60° was used before the activation of the optimisation process. A small step in the bank angle at 10 seconds into the flight can be seen in Figure 12. This indicates that a lower bank angle in the flight region before the optimisation began would have been sufficient.

In the region of 145 seconds, there is a peak in the angle of attack parameter, before it steadily reduces to a constant value of 4°. This increase was produced to account for the loss of lift due to the high bank angle, which was at a value of -58° in this flight region.

Comparison

After the two vehicle concepts had been optimised, the vehicles were compared on a performance and operational basis to determine which vehicle would best suit commercial requirements. It was decided not to refine the optimum trajectories any further as the changes made were found to have little effect on the payload placed into orbit.

Powered return concept

As was seen previously, a higher staging velocity resulted in a higher payload to orbit. Because of this, the powered return flight concept vehicle was found to be capable of delivering a 50% higher payload to orbit than the un-powered return concept vehicle.

A powered vehicle would also have significantly more operational flexibility and would be able to land at a range of other sites in the case of an emergency. This would lead to improved safety for the mission as well as reduced risk of losing the vehicle and payload in the event of the vehicle not being able to land at the launch site.

The initial cost of this concept vehicle would be higher than that for the un-powered return vehicle concept, due to the added hardware and complexity of air breathing engines. The operational costs would also be higher due to servicing and maintaining the air breathers. It is, however, believed that the higher income due to the improved payload capability would outweigh the extra component and operational costs.

Un-powered return concept

This concept is likely to have lower vehicle costs as well as lower operational costs, due to the absence of air breathing engines on the booster. There would however be considerable logistical limitations.

The full glide potential of the vehicle would not be exploitable due to the safety implications. A considerable factor of safety would have to be applied to the fly back distance in case the vehicle encountered unexpected weather conditions or some other external influence. The vehicle would also be limited by the number of landing sites it could fly to in the event of the launch site not being available for landing. The most significant disadvantage of this concept is its inability to exploit high staging velocities, which support higher payload capabilities. This is due to the limited fly back capability of the booster.

Design choice

When considering the two concepts discussed in this paper, it was decided to choose the vehicle employing air breathing propulsion for the return flight. Although this vehicle would be more complex and have higher operational costs, the payload advantages seem to be significant enough to make it the preferred design concept. For servicing GTO missions, it is recommended that the vehicle use an expendable third stage. Table 4. shows the mass distribution for the powered return flight concept vehicle .

Description	Booster	Orbiter
Base structure and tanks [tons]	22,2	8,3
Aerodynamic surfaces [tons]	3,7	6,1
TPS [tons]	4,4	[illegible]
Propulsion [tons]	13,7	[illegible]
Auxiliary systems [tons]	14,7	10,3
Uncertainties [tons]	4,9	3,3
Dry mass [tons]	63,6	39
Propellants [tons]	1039	209
Payload [tons]	277	21,6

Table 4. Mass break down

Conclusions

It can be concluded that the optimal staging conditions for a 2 stage reusable vehicle is at approximately 3000m/s for a powered return flight and approximately 1200m/s for an un-powered return flight. Further, the staging velocity for an un-powered fly back concept should be as high as possible while still low enough to allow a safe return glide flight to the launch site.

The investigation also indicated that for a staging velocity of 3000 m/s, it would not be advantageous to employ stronger and heavier wings to allow for a high load turn during fly back. The added wing weight would outweigh the payload gains obtained by a high load turn. It is thus beneficial to design the wings for loads encountered during landing and then limit the loading during the fly back manoeuvre.

It is the view of the authors that the optimal design for a reusable launch vehicle would be to employ a powered fly back concept rather than an un-powered. There would be additional costs and complexity with the powered vehicle due to the inclusion of air breathing engines, however, it is believed that these would be compensated for by the significantly improved payload capability. A further suggestion is the use of an expendable upper stage vehicle to achieve GTO. This would limit the mass that needed to be accelerated to orbit thereby increasing the GTO payload capability.

Acknowledgments

We would like to thank The Sir Ross and Sir Keith Smith Fund for providing funding for the research. We would also like to thank the Space Systems Institute, Stuttgart University for the use of their software and computing services.

References

[1] S.J Isakowitz, International Reference Guide to Space Launch Systems, AIAA, 1995.

[2] B.S Middleton, The Global Space Launch Market in the Next Decade, International Aerospace Congress, 1999.

[3] Daimler-Benz Aerospace, FESTIP system study, August 1998.

[4] D.C Freeman, T.A. Tally, D.O. Stanley, R.A Lepsch and A.W. Wilhite, Design Options for Advanced Manned Launch Systems, Journal of Spacecraft and Rockets, Vol 32, No 2, March-April 1995, 241-249.

[5] NASA web site, X-34 concept vehicle.

[6] M. Rahn, U.M. Schöttle, E. Messerschmid, Impact of mission requirements and constraints on conceptual launch vehicle design, Aerospace Science and Technology, 1999, no. 6, 391-401.

[7] S.P Chakroborty, C. Limerick, R. Stark, A systematic Approach for the Development of the Propulsion System for the K-1 Reusable Launch Vehicle, IAF-98-s.1.08 49th International Astronautical Congress, Melbourne, Australia, 1988

[8] U.M. Schöttle, Mission Dependend Selection of Propulsion Systems for Reusable Launch Vehicles, Second European Aerospace Conference on Progress in Space Transportation, Bonn-Bad, Godesberg, Frg, May 1989.

[9] R.W. Powell, J.C. Naftel and C.I. Cruz, Ascent Performance Issues of a Vertical-Takeoff Rocket Launch Vehicle, Journal of Spacecraft, Vol.28, No.2, March- April, 1991, 179-183.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37240

AIAA-2000-4181

THE LIFE CYCLE APPLICATION OF INTELLIGENT SOFTWARE MODELING FOR THE FIRST MATERIALS SCIENCE RESEARCH RACK

Amanda Rice – Raytheon Information Technology and Scientific Services
Frank Parris and Philip Nerren – Sverdrup Technology, Inc.
620 Discovery Drive
Huntsville, Alabama 35806

Abstract

An intelligent software model of First Materials Science Research Rack (MSRR-1) is under development and will serve multiple purposes to support the engineering analysis, testing, training, and operational phases of the MSRR-1 life cycle development. The G2 real-time expert system software environment developed by Gensym Corporation was selected as the intelligent system shell for this development work based on past experience gained and the effectiveness of the programming environment. We are adopting a multi-purpose approach with our G2 model by employing its intuitive graphics and robust natural language programming cability to provide an engineering environment for rapid prototyping analysis, to provide simulators, and to develop real-time command and control systems.

History

Marshall Space Flight Center (MSFC) has been funding development of intelligent software models to benefit payload ground operations for nearly a decade. Experience gained from simulator development and real-time and control monitoring of NASA Spacelab missions is being applied to engineering design, testing, and operation of the MSRR-1. MSRR-1 is the first rack in a suite of three racks comprising the Materials Science Research Facility (MSRF) which will operate on the International Space Station (ISS). The MSRF will accommodate advanced microgravity investigations in areas such as the fields of solidification of metals and alloys, thermo-physical properties of polymers, crystal growth studies of semiconductor materials, and research in ceramics and glasses. The MSRR-1 is a joint venture between NASA and the European Space Agency (ESA) to study the behavior of different materials during high temperature processing in a low gravity environment. The planned MSRR-1 mission duration is five (5) years on-orbit and the total design life is ten (10) years. The MSRR-1 launch is scheduled on the third Utilization Flight (UF-3) to ISS, currently in July of 2003. The objective of MSRR-1 is to provide an early capability on the ISS to conduct material science, materials technology, and space product research investigations in microgravity. It will provide a modular, multi-user facility for microgravity research in materials crystal growth and solidification. The initial launch configuration supports investigations in one side of the rack by commerical agencies, and the other side of the rack accommodates the ESA Materials Science Laboratory (MSL) which accommodates ESA and US sponsored furnace investigations.

MSRR-1 Systems Overview

The MSRR-1 consists of three basic subsystems, which provide ISS resources to two furnaces, or experiment modules (EMs), for scientific processing. The EMs are designed and integrated with their own secondary subsystem control. These secondary controls self regulate each furnace to a finer degree based on their individual resource loading requirements. Although each EM can operate independently, an interleaved dual operation concept has been developed for optimization. Therefore, the rack facility is responsible for distributing and arbitrating these resources to each EM system as they may operate in parallel, as long as station resource constraints such as power and cooling capability, are not violoated. The basic resource allocations defined for the operation of these two material science furnaces consist of:

- Power and communications management systems - These resources are provide by station and managed by two rack subsystems: the Solid State Power Control Module (SSPCM) and the Master Controller (MC). The SSPCM controls and distributes power to all hardware in the MSRR-1 rack. The MC provides a communications interface to station and oversees and relays commands and telemetry data. Data communications are

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

achieved with 1553B, Ethernet, and video buses.

- Primary and secondary fluid cooling loops – The Thermal Control Environment System (TECS) uses a primary water coolant loop which is interfaced with space station fluid lines to maintain a closed loop system. The furnaces reach high temperatures during processing, so the TECS water loop removes waste heat, keeping rack temperatures within safety limits. Each EM manages independent, secondary, closed fluid loops. These two independent loops are interfaced to the primary loop via a heat exchanger.

- Vacuum Access System (VAS) – The VAS provides independent accessibility to a space vacuum depending upon operational requirements. Furnace operations require processing of samples at a high temperature in a vacuum or an inert gas environment. Vacuum exhaust allows only one user at a given time and is needed to purge the atmospheric volume around the furnace. This singularity of access allows a payload to purge waste gases to achieve a basic vacuum of about 10^{-3} torr. Once this crude vacuum is established then the system can be switched, if desired, to the resource lines, for achieving and maintaining better vacuum levels. The resource line allows up to six users simultaneously. This line provides a vacuum levels significantly lower depending upon the parallel users in the system. The rack subsystem has isolated valve control for each type of vacuum access to prevent cross contamination of the furnaces during dual operations.

The MSRR-1 subsystems and EM locations are shown in Figure 1.

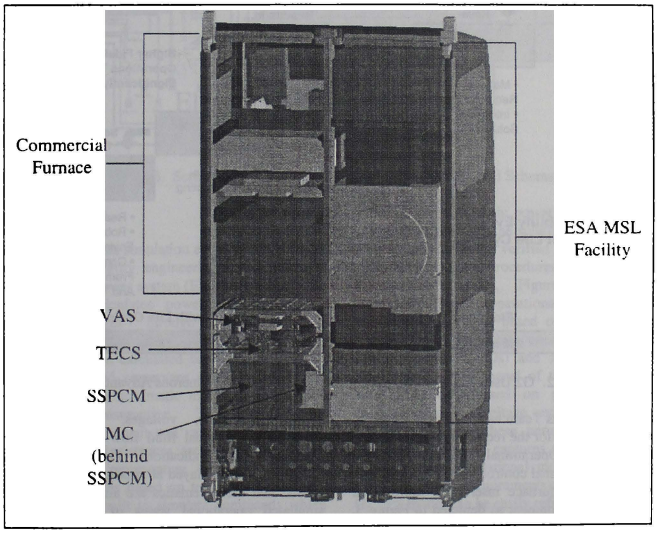


Figure 1. MSRR-1 Initial Rack Configuration

Life Cycle Application of Intelligent Software Modeling

Figure 2 depicts the traditional life cycle development phases of an aerospace program using G2, starting with conceptual design of a system, progressing through requirements development, design, fabrication and testing, and the training and operations phases. The benefits of implementing the G2 environment through each life cycle phase are listed. Initially, G2 offers a rapid prototyping capability to model system concepts and examine the impacts of differing system design options. Within the requirements phase, modeling derives and verifies mature requirements, adding confidence to established requirements. As the design progresses, G2 software bridges can be built to link G2 with traditionally used detailed design tools for analyzing system thermal, electrical, and environmental characteristics. For fabrication and test, bridges can also be built to link G2 with hardware components; essentially substituting some G2 models within the environment with hardware test behavior. Finally, mature G2 models provide realistic training prior to real-time execution of the mission, and the familiar model environment can be used to monitor the flight systems. Additionally, the G2 environment offers numerous advantages for increased software development efficiencies such as: an intuitive and flexible user interface, and use of a natural language editor for defining the object hierarchy and entering inferencing knowledge; no extensive knowledge of programming languages is required.

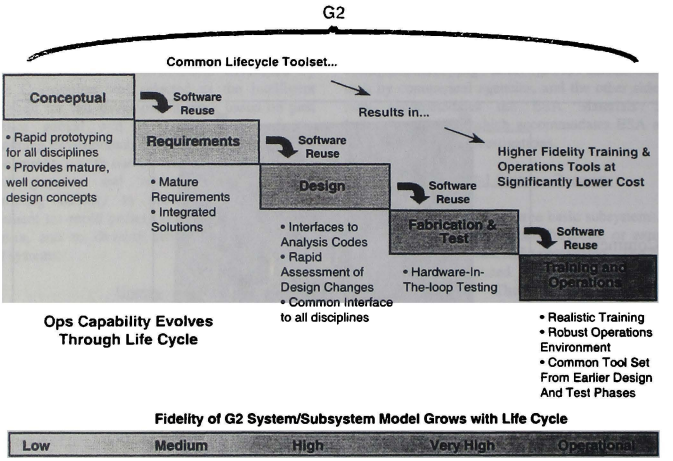


Figure 2. G2 Used Across a Program's Life Cycle Provides Numerous Advantages

The MSRR-1 Operations Team has developed an initial model which is tailored for the requirements and design life cycle phases. Operational schematics of the MSRR-1 electrical, thermal control, vacuum access /gas supply systems, and furnace inserts are represented graphically in the environment as depicted in Figure 3. Logic to represent first order engineering calculations is coded into the knowledge base to simulate the operational behavior of the MSRR-1 systems. An example of engineering data provided includes electrical currents, voltages, operational power, temperatures, thermal fluid flow rates, pressures and component status indications. These types of data are calculated and displayed at appropriate instrumentation points, and the schematics are animated to reflect the simulated operational status of the MSRR-1. The software control functions are also simulated to represent appropriate operational behavior based on automated control and response to commands received by the crew or ground controllers.

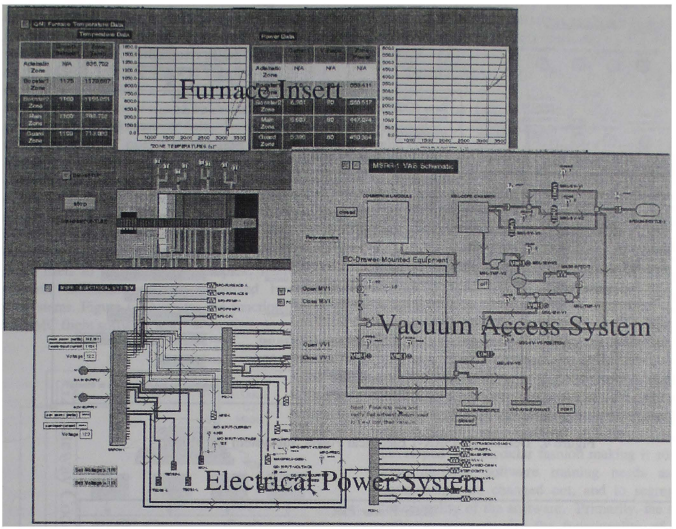


Figure 3. Screen Shots of MSRR-1 G2 Subsystem Operational Schematic

Engineering Analysis

The first benefit of this simulation environment is being realized in high fidelity engineering analysis results from the electrical power system (EPS) G2 model. The EPS simulation calculates power requirements by propagating voltages through each electrical component. The current requirements (amps) for each end item load is then calculated based on load power values. The required amperages are then propagated back up through the schematic to derive the overall MSRR-1 power consumption. The observed load power for each component will vary depending on its operational state. Resistances are accounted for and even the equation representing the primary power distribution box losses based on test data is reflected. Buttons have been built to simulate commands which power up loads as it is expected in the real-time profile to derive overall system power profiles. Additionally, power allocations for the subsystem equipment and the furnace inserts can be verified based on the simulated operational profiles. Procedures can easily be written to represent duty cycles. Figure 4 captures modeling screens showing the operational panel which activates and deactivates loads based on the planned on-orbit cycling of hardware components, the simulated light emitting diodes (LEDs) and switches which will be present on a front panel of the rack and how these lights illuminate based on power status, and an enlargement of the voltage and current sensors which will be simulating these parameters for the Master Controller. Figure 5 captures an electrical power plot during planned operational modes. As test data for hardware power requirements becomes available, the G2 EPS model can be easily refined.

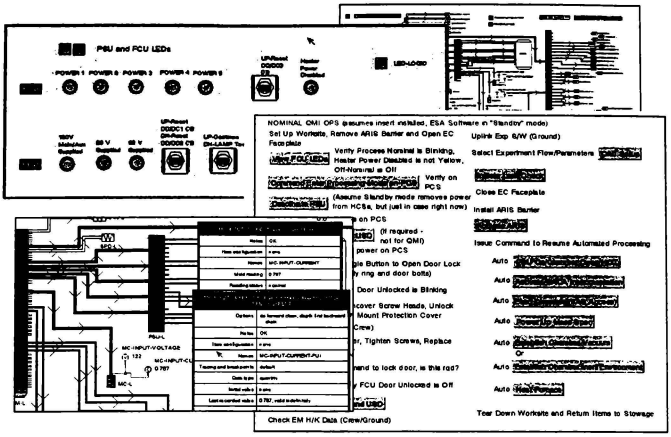


Figure 4. Details of MSRR-1 Electrical Power Subsystem G2 Model

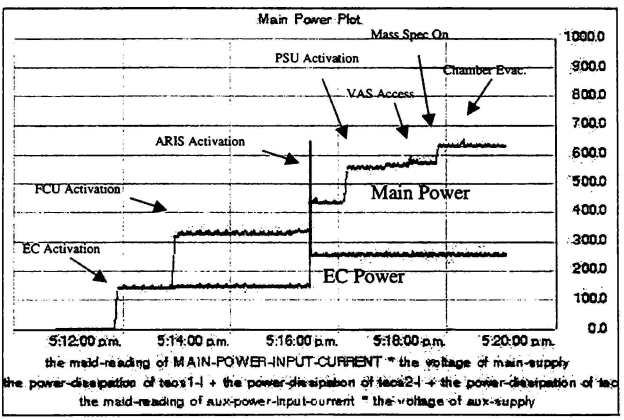


Figure 5. Simulated MSRR-1 Power Plots

Hardware In the Loop Testing

Our G2 model will also be used for hardware in the loop testing. The MSRR-1 model will be augmented with additional software as part of the test configuration of the MSRR-1 MC. We will take advantage of G2's capability to simulate a flight like data stream to test flight software responses and behavior. The G2 model provides a reusable test module, developed to closely mirror subsystems' operating characteristics, so that the MSRR-1 MC hardware and flight software can be accurately tested prior to integration with other rack components. Total system dynamics can be altered to account for a variety of expected operating conditions, including both those imposed by external ISS conditions, as well as varying internal conditions. G2 lends itself to modeling the complex, interdependent resource utilization scheme employed by MSRR-1 subsystems. Figure 6 illustrates the functionality of G2 with MC flight software test.

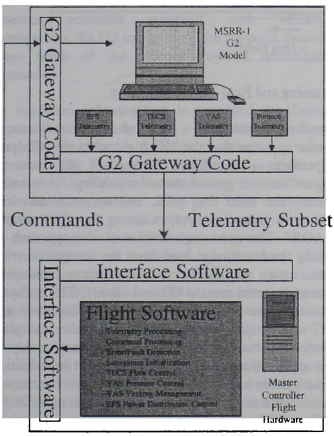


Figure 6. Master Controller and G2 Testing

Training Simulation

The MSRR-1 G2 model will support training of the astronauts who will conduct on-orbit MSRR-1 operations and the ground support personnel (GSP) who will monitor payload operations via telemetry. The MSRR-1 software simulation model will complement a hardware mock-up, called a Payload Training Unit (PTU), of the MSRR-1 to provide crew training on integrated payload operations as shown in the lower left ISS payload rack representation of Figure 7. The MSRR-1 G2 model will be located on a personal computer located at the lower back of the simulated MSRR-1 rack at the JSC Payload Training Complex (PTC). The G2 model will be controlled from an Instructor Operator Station (IOS) which can monitor and control numerous distributed G2 models within the training facility. G2 gateway code will output the simulated MSRR-1 instrumentation values in a flight-like data stream so the crew has realistic, accurate simulated MSRR-1 data on flight displays designed for crew use. The simulation will also respond appropriately to crew or ground initiated commands, which will be part of normal facility operations. The G2 model also provides the capability to implement signal conditioning equipment (SCE) to drive mechanical interfaces that are part of the MSRR-1 model. Operational mechanical equipment on the training hardware which will be driven by G2 include light emitting diodes (LEDs) and switches, provided to give insight to ESA MSL functions, door locks, and a low fidelity motor to simulate motion of furnaces during processing. The training simulation software is developed in a modular fashion making it adaptable to long duration software training needs as payload components are changed out, and to segregate basic functionality of the software. Primarily, the simulation software accounts for the functionality of the Rack Support Systems, the elements of the two Experiment Modules (EMs) which are integrated into each side of the MSRR-1 rack, and the furnaces and samples which contain the processing material to be melted and solidified for scientific analysis. The ability to establish relationships in G2 to link objects in one subsystem with related objects in another subsystem assures integrated system behavior. For example, the Master Controller computer is activated in the EPS model at a certain power which also implies power dissipation of the Master Controller. A Master Controller object is also represented in the TECS model. The power dissipation of the Master Controller in the EPS model is automatically passed into the TECS model as Master Controller heat dissipation, and the TECS knowledge base of rules update the TECS water loop temperature.

The G2 model will also be used to train the Ground Support Personnel who will monitor the MSRR-1 systems and payloads while they are operating aboard the ISS. The scope of the operational scenarios simulated will be expanded to include additional functions in which ground actions will be necessary, such as diagnosis of a component failure.

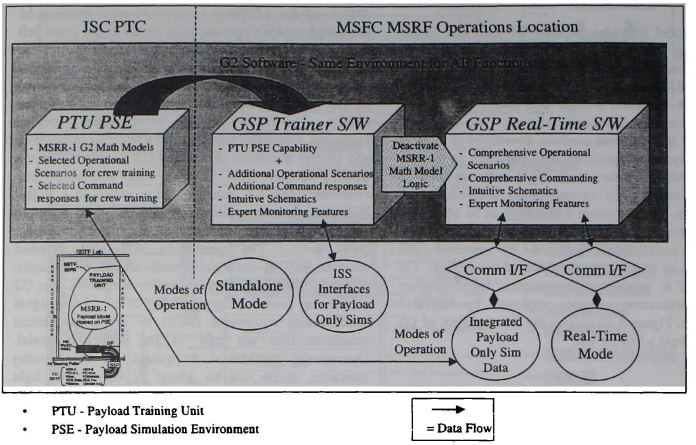


Figure 7 - use of MSRR-1 G2 Model for Training and Real-Time Monitoring

The intuitive, schematic based environment will provide an excellent foundation for personnel to understand the integrated configuration and operation of the MSRR-1, and the anticipated telemetry feedback based on operational modes of the equipment.

Expert monitoring features will be enhanced to provide a smart monitoring environment for the operators. These features include:

- Animated, intuitive schematic-based displays which reflect telemetry values
  Real-time plotting of simulated or incoming sensor values
- High/Low exception monitoring for analog data
- Expected state monitoring for discrete data
- Data trending
- Automated malfunction procedure execution to diagnose problems

Real-Time Monitoring

Real-time monitoring of the MSRR-1 payload poses a challenge for operations engineering development. Constraints on ISS payload operations include minimal crew availability, short periods of telemetry acquisition of signal (AOS), limited rack resources, and limited ground personnel. As a result of these conditions, it became apparent that the MC flight software will have to perform a significant amount of the tasks in conjunction with ground control capability. This reality dictated the need for quick and accurate decision-making by ground controllers during the short periods of AOS. The operations team took the approach of shifting the knowledge base from the design/operation ground team members to the real-time monitoring systems centered around G2 and other COTS software solutions. The goal is to develop an intelligent system that would aid in operational analysis and data processing, management, distribution and analysis. This approach requires a significant effort in definition of the system architecture to achieve these operational goals. These goals are achieved through COTS hardware and software solutions.

G2 will be the foundation for the real-time monitoring and control environment software. For long term missions, G2 allows one to capture expert knowledge in the form of rules or procedures. This allows the inference engine to build on the knowledge which can be used to monitor systems in a reliable and repetitive way. Reuse of G2 analysis and simulation models used earlier in the MSRR-1 life cycle will be implemented.

The same schematic environment will interface to the incoming data for the real-time telemetry stream to represent the onboard hardware configuration. This is achievable by deactivating the logic which was used to calculate telemetry values, and replacing telemetry values with live telemetry. And finally, the logic will be added to the monitoring environment over time which replicates the ground controllers thought process of determining MSRR-1 health and responding to actions. Some major advantages that G2 offers beyond an intuitive, visually-oriented environment include: the ability to easily discern trends in data and perform trend analysis; and the capacity to operate on diverse information sources, inferring knowledge of system status, and applying these inferences through messages to an operator or real-time control of systems. Scientists remotely located from the real-time control center may access real-time data in a number of ways. One implementation includes the G2 companion product, Telewindows, which allows remote windowing into the host G2 running on the G2 peripheral master host via network or modem connectivity. With numerous scientists involved with MSRR-1 investigations, the G2 implementation supports dynamic access needs of the science Payload Investigator community.

The hardware infrastructure is provided by Ziatech Corporation. Their bus arbitration software CompactNET makes it possible to develop a robust system. The CompactNET software provides the necessary arbitration and multitasking capabilities for multi processing systems. The goal is to configure a compact PCI multi processor system in such a manner that each processor board is assigned a specific function in the operational scheme. Because these multiprocessors are linked across the same bus backplane, the arbitration software can share data and tasks between each processor. An illustration of this configuration can be seen below in Figure 8. The network of processors makes it possible to distribute large amounts of telemetry data for manipulation and interpretation. With current processing speeds and a distributive task concept, complex analysis, simulations and data processing can be achieved much quicker. It is our plan to receive the raw telemetry data from space station via the Payload Operation Integration Center (POIC) directly into the main processor, which is configured as a system master (SM). This system master receives the raw data and processes from encrypted CCS data packets into formatted engineering units. The system master interprets the formatted data stream and distributes the necessary information to their designated peripheral master (PM) processor for task specific data manipulation. The tasks have been broken down by operations team function in the Table 1. By linking and networking these processors, all of the ground control functions can be operated and monitored from any given workstation. This provides redundancy in the event of a processor failure. Another feature of this redundancy is the issue of reducing manpower. The multitasking and redundancy of this architecture carries the bulk of the knowledge freeing the operator to manage the now intuitive information and not processing intricate data.

Summary

The MSRR-1 G2 simulation model spans many elements of the life cycle development of this project: Engineering Analysis, Test and Checkout, Training of Crew and Ground Personnel, and Real-time monitoring and control. By utilizing the unique features afforded by an expert system development environment, we have been able to synergize a powerful tool capable of addressing our project needs at every phase of project development. Through the use of an intelligent system software development environment, we are able to accurately model the diverse components of our rack and to analyze interactions during various design phases. Having a reuseable core software module ensures consistency in test results, training, and application to flight operations. Resulting benefits to the project include cost and time savings, more reliable software, more easily maintained software, and flexibility in every aspect of the G2 applications for system design and operation. To realize significant reduction in payload operations costs, intelligent systems provide an avenue to tirelessly and accurately monitor data, to provide an effective interface which can maintain interest for the operator, and to provide tools which are easily expandable and adaptable.

References

1. Robert C. DaLee, Amanda B. Rice, and Mark N. Rogers, "Intelligent Software Tools for Enhanced Spacecraft Payload Ground Operations at NASA/MSFC", AIAA-97-3934, AIAA Defense and Space Programs Conference and Exhibit, Huntsville, Alabama, September 23 – 25, 1997.
2. MSRR-1 System Description Document, MSRR-1-DOC-0100, September 3, 1999.
3. Payload Training Unit Development Specification, MSRR1-SPEC-0015, April 3, 2000.
4. Operations Concept Command and Control Plan, MSRR1-DOC-0001, April 3, 2000.
5. Electrical Power and Energy Management Report, MSRR1-EPEMR-001, March 23, 2000.

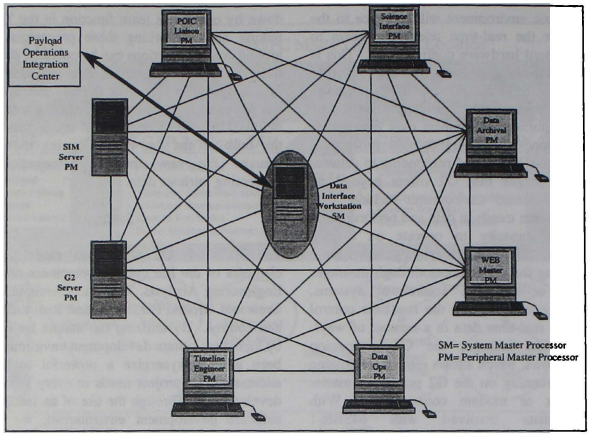


Figure 8. Real-Time Architecture Implementing G2

Table 1. MSRR-1 Real Time Operations Team Functionality and Supporting Software Solutions

Workstation User	Function	Software/Data Products
POIC Liason	Monitors all system operations and checkout (i.e. video system). Supports anomaly resolution.	G2 /engineering data from system and components. TECS,VAS,SSPCM/ Timeline event viewer
Science Interface	Monitors and analyzes experiment data. Possibly monitor/command ground based experiment simulations in G2.	Labview/ plotting temperature profiles, acceleration data. G2/ Heater performance/characterization.
Data Operations	Monitors health and status, responds to error messages, supports troubleshooting in real-time. Data/Video processing and reduction. Sends commands to the facility. Enable/disable remote command users	Manual Procedure Viewer./Procedures Labview/Image processing, data decommutation G2/ system engineering data, command formulation and implementation
Timeline Engineer	Procedure updates (ISS automated & crew), crew support, POIC Operations Control team interface, initiates & tracks Operations Change Requests (OCRs). Responds to POIC replans, works with POIC Planning Team to plan operations and resources.	Process/ Resource analysis and simulations.
Web Master	Develops/maintains website for data dissemination to remote payload and science team members.	HTML/ Science plots, images and reports G2/ Telewindows operations to remote users
Simulation/Training Coordinator	Develops/maintains/conducts simulations during pre mission and mission operations.	G2/ payload operations displays

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37241

AIAA-2000-4182

CONTROL BOOM DESIGN OF Z-SAT

K. Daneshjou*, A. Alasty†, S. Rezaei‡

Advanced Electronic Research Center (AERC) and
Sharif U. of Tech. (SUT), Dep. of Mechanical Eng.
Tehran, IRAN

Abstract

Z-SAT uses passive gravity-gradient, along with active magnetorquer stabilization methods to control of attitude. To modify the ratios of moments of inertia, control boom has been selected. It is a deployable mechanism. Several restrictions, such as stability, strength, and thermal distortion, have been considered to optimum design. In this paper, the relevant design process is described. Proper configuration of the boom is determined ; then the stability analysis is done on the satellite (along with boom), to make sure that the system would deliver the right response. Combination of analyses mentioned above results in optimum geometrical, physical, and timing characteristics of control boom of Z-SAT.

Introduction

Control boom is the inseparable member of gravity-gradient stabilization method. Using the boom, distribution of mass moments of inertia is optimized to take most advantages of gravity field of earth. Related to both attitude control and deployable mechanism of the boom, various configurations have been developed. Mr. Hughes has reviewed conventional configurations of the boom[1]. In the other research, Mr. Conely has described different types of deployable mechanisms[2].

More than modifying the distribution of inertia moments, control boom is also used as a damper. In this case, boom has some degree-of-freedom related to main body of the satellite.

Strength and thermal considerations are also the significant problems of booms. During deployment of the boom, different loads, such as gravity and drag, enforce it; this may cause damage of the boom[3]. After deployment, temperature of the boom varies because of variation sun radiation angle. These thermal variations sometimes result in severe attitude delibration[4].

Transient deployment time has a high importance. Mr. Connell by minimizing a cost function containing attitude errors designed ideal (time-variant) configuration[5]. However, it has some practical restrictions.

In this paper, process of design of control boom of Z-SAT is described. Its boom is similar to one depicted in by Mr. Etkin[6]; with this different that has no degree-of-freedom. Control, strength, and thermal considerations have been heed in this process.

Configuration of Z-SAT

This satellite is from the series of micro satellites and is to move in a near earth circular orbit of 700 km. altitude. Satellite is axisymmetric and gravity-gradient stabilization method has been proposed for that. Details of the attitude control system have been described in literature[7]. Schematic configuration of this spinning satellite is shown in figure (1).

Length of the boom is some 7 meter and its tipmass has an equivalent mass of 2.6 kg.. Pre-deployment mass moments of inertia are about 1 and 2 kg.m^2 for axial and transverse directions.

Control Considerations

To achieve the most gravity-gradient torque, it's necessary to have a narrow and long symmetrical satellite. On this purpose, a mass of 2.6 kg. has been allocated at height of 7 m above the satellite surface. This mass increases the transverse moment of inertia up to 134 kg.m^2, after deployment of the boom. From control viewpoint, more the length and tipmass, more better. However, strength conditions do not permit this.

The control boom of Z-SAT is deployed at a certain attitude. Attitude of the satellite is defined using deviation angles, Pitch and Roll (figure (2)). Pre-deployment values of pitch and roll angles have been selected 60 and 30 degree, respectively.

These deviations are provided using magnetorquers. Magnetorquing is the active stabilization method of Z-SAT. In figure (3), during-deployment variation of pitch angle versus roll angle has been drawn. Proper pre-deployment attitude has caused this smooth behavior. Deployment time is 15 min.. Boom is deployed on a constant rate. Figure (4) shows post-deployment variation of deviation angles. Optimum value of period of the axial rotation is 5 min. Having no severe delibration in attitude is in the reason of proper mass moments of inertia distribution.

Mechanism and Strength Considerations

Deployable mechanism of the boom is motorized telescoping type. Among different types of booms, including tubular, telescoping, coilable, and

---

* Head of Structural Department of AERC, also Assistant Professor, Dep. of Mech. Eng., IRAN University of Science and Technology.
† Assistant Professor, Dep. of Mech. Eng., SHARIF University of Technology.
‡ Senior Researcher of AERC, also MSc. of Mech. Eng., Dep. of Mech. Eng., SHARIF University of Technology.
§ Copyright © 2000 The American Institute of Aeronautics and Astronautics Inc. All rights reserved.

articulated ones[2], this type has been selected because of its small-stowed volume, and good ability on enduring external loads. Major strength considerations are related to transient deployment time of the boom. In this phase, boom has not its ultimate strength. Drag and gravity are external loads in this phase. Values of these orthogonal loads are as follows,

$$F_D = 205\ nN \quad (1)$$

$$F_g = 20.7\ N \quad (2)$$

In which, F_D is the drag force and F_g is the gravity force.

Mr.'s Ahmadi and Ashtari have described details of the strength analysis[8]. To endure bending caused by these loads, large diameter of the boom must be near 400 mm. More increasing length of the boom, more this diameter must be.

Thermal Considerations

Of the major advantages of axial rotation is removing of thermal gradient across the boom. Variation of mean temperature of the boom is not also so severe, in the reason of satellite fast motion. Figure (5) shows variation of temperature versus time (in an orbital period). As shown, difference of maximum and minimum temperatures is less than 50 centigrade, which also occurs, in a long time.

All after, effects of thermal shock and distortion on Z-SAT are negligible.

Conclusion

In this paper, assumed considerations in design of control boom of Z-SAT was depicted. These considerations include control, strength, and thermal problems. From the control point of view, using a 7-meter boom and allocation of a 2.6-kilogram mass on its end optimized distribution of mass moments of inertia very much. These distributions minimized delibration of attitude during and after deployment of the boom. The strength analysis didn't permit longer length of the boom. This analysis selected telescoping mechanism. Related to thermal shack and distortion, no severe problem was found; because, the axial rotation removes thermal gradient across the boom.

Acknowledgment

Advanced Electronic Research Center of Iran has sponsored this research.

References

[1] Connell, G.M., "Degree-of-Freedom Requirements for Hinged Multibody Satellites," *Journal of Spacecraft and Rockets*, Vol. 6, No. 1, pp. 90-92, 1969.

[2] Conely, P., *Spacecraft Mechanisms*, John-Wiley & Sons, 1998.

[3] Hughes, P.C., *Spacecraft Attitude Dynamics*, John Wiley & Sons, 1986.

[4] Connell, G.M., Chobotov, V., "Possible Effects of Boom Flutter on the Attitude Dynamics of the OV1-10 Satellite," Journal of Spacecraft and Rockets, Vol. 6, No. 1, pp. 90-92, 1969.

[5] Connell, G.M., "Optimal Configuration for Hinged, Two-Body Satellite," *Journal of Spacecraft and Rockets*, Vol. 6, No. 9, pp. 1024-1030, 1969.

[6] Etkin, B., "Dynamics of Gravity-Oriented Orbiting Systems with Application to Passive Stabilization," *AIAA Journal*, Vol. 2, No. 6, pp. 1008-1014, 1964.

[7] Rezaei, S., "Dynamic Modeling and Stability Analysis of a Sample Satellite with Considerations of Earth Orbit Effects," *Sharif University of Technology*, Department of Mechanical Engineering, M.Sc. Thesis, Feb. 2000, (In Persian).

[8] Ahmadi, S., Ashtari, H., "Strength Design of Z-SAT Control System," *Iran University of Science and Technology*, Department of Mechanical Engineering, B.Sc. Thesis, Feb. 2000, (In Persian).

Figures

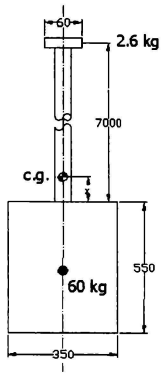


Figure 1, Schematic configuration of Z-SAT

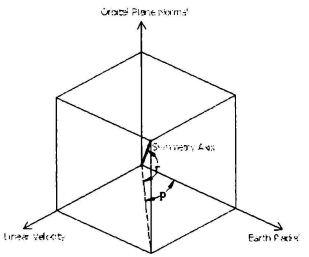


Figure 2, Deviation angles: pitch and roll

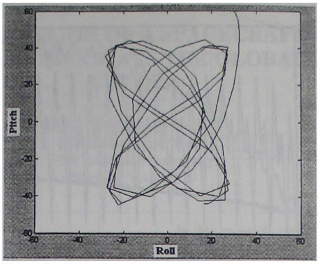
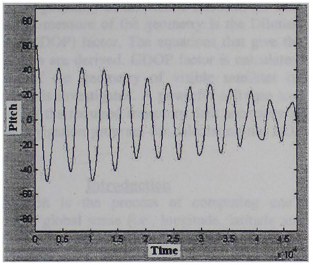
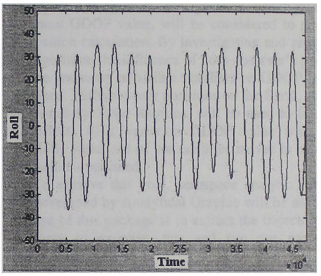
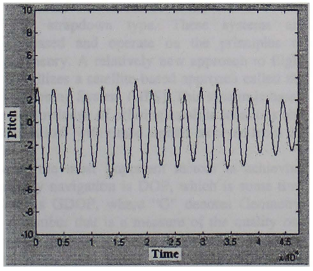


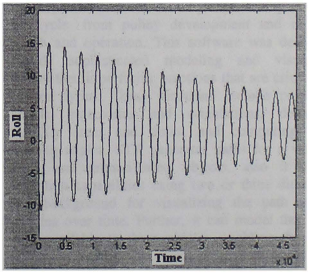
Figure 3, During-deployment variation of pitch angle versus roll angle





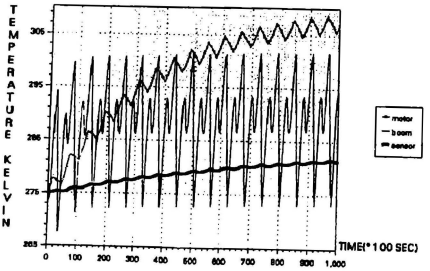
Start Time=0 ; Finish Time=790 min.





Start Time=3160 min. ; Finish Time=3950 min.

Figure 4, Post-deployment variation of pitch angle versus roll angle



Figures 5, Variation of mean temperature of the boom versus time

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37242

AIAA-2000-4184

SET UP AND UTILIZATION OF A SPACECRAFT MECHANICS SOFTWARE TOOL TO SIMULATE THE GLOBAL POSITIONING SYSTEM

Abdul Ghafoor Al-Shehabi*
Aerospace Department
Old Dominion University
Norfolk, VA, 23529

Abstract

The Global Positioning System (GPS) is the latest technology in navigation and location determination. The principal behind GPS is the measurement of distances or ranges between the receiver and the satellites. Due to the relative motion of satellites, the distances and geometry of the visible satellites are time dependent. A measure of the geometry is the Dilution of Precision (DOP) factor. The equations that give the user's position are derived. GDOP factor is calculated. The effect of the geometry of visible satellites on GDOP factor is investigated. A powerful software tool is used to get the required trajectory data necessary as input to a navigation program for calculating the DOP factor.

Introduction

Navigation is the process of computing one's position, in a global sense (i.e., longitude, latitude and altitude), from physical measurements and fundamental governing principles. This process is an essential component in modern flight systems, specifically to provide feedback signals for trajectory guidance systems. Conventional flight navigation systems are based on inertial measurement units, either of the platform or strapdown type. These systems are mechanical-based and operate on the principles of gyroscopic theory. A relatively new approach to flight navigation utilizes a satellite-based approach called the Global Positioning System (GPS). This system is based on precise knowledge of orbiting satellite locations and "triangularization" with timed signals received from the satellites

One of the most important factors in achieving high accuracy navigation is DOP, which is some time referred to as GDOP, where "G" denotes Geometry. DOP is a number that is a measure of the quality one might expect to achieve from a position measurement with the GPS system based solely on the geometric arrangement of the satellites used in the measurements.

In this research, calculating GDOP factor and other factors related to DOP will be introduced by linearizing the pseudorange equation. This linearization represents the changes in user position and time bias. An algorithm by which, the receiver can determine which is the optimal group of four satellites in view will be developed. The optimum configuration, which gives the minimum GDOP value, will be considered to perform the position calculation. By investigating and analyzing this algorithm, the influence of both observer's position and the geometry of visible satellites on GDOP will be explained. Different user positions on the Earth, such as Equatorial vs. Polar Regions, will be used to illustrate this effect. Also, variation GDOP value with time, especially when one of the satellites moves in or out of view will be out lined.

To achieve this goal, aerospace analytic software tool developed by Analytical Graphic will be used. The purpose of this package is to extract the trajectory data necessary to exercise, test and run our navigation algorithms.

Satellite Tool Kit

Satellite Tool Kit is a complete set of advanced software tools that address all phases of a space system life cycle, from policy development and design to launch and operation. This software was designed to provide sophisticated modeling and visualization capabilities, it performs functions that are critical to all mission types. These functions include propagating vehicle trajectories and orientations, determining visibility areas and times, and computing sensor-pointing angles. STK generates paths for a variety of space and ground-based objects, also it provides animation capabilities using two or three dimensional map background for visualizing the path of these vehicles over time. Further, it can model the field of view available to these objects, enabling the user to determine whether the object can see its target or not. Users are able to display and analyze satellite, aircraft, ship, and land vehicle paths, and utilize reporting and graphic capabilities provided by STK at both the workstation and PC to help them to communicate results effectively.

* Ph. D. Candidate. Student Member AIAA

Copyright © 2000 The American Institute of Aeronautics and Astronautics Inc. All rights reserved

Global Coordinate Systems

North-East-Down Coordinate System

The North-East-Down coordinate system is the most commonly used system today to determine position of any body on the Earth surface. It moves with the body, so that the x-y plane is tangent to the Earth surface. These coordinates are referred by NED, where N-axis points toward the North Pole, D-axis points toward the center of the Earth, and E-axis completes a right handed orthogonal system and points toward East. The orientation of Cartesian coordinate (NED) changes with the time relative to the center of the Earth, its position is determined by two angles and the distance between the location and the center of the Earth. The Prime Meridian and the Equator are the reference planes used to define the two angles latitude and longitude. Longitude angle θ is measured in degree east and west of prime meridian, which is defined to pass through the site of the Royal Observatory in Greenwich, England. Latitude angle λ is measured in degrees above or below the Equator.

Earth Centered Earth Fixed Coordinate System

The Earth Centered Earth Fixed coordinate system is used to define three-dimensional space relative to the earth. The Cartesian coordinates (XYZ) define three-dimensional positions with respect to the center of the earth. Also here, ECEF coordinates are referred by XYZ. X-axis is defined by the intersection of the plane defined by the prime meridian and the equatorial plane. Z-axis points toward the North Pole, while Y-axis completes a right handed orthogonal system by a plane 90° east of the X-axis and its intersection with the equator.

The Relationship Between NED and XYZ

Since the position coordinates for both satellite and receiver are given in local NED coordinates, the transformations for both satellite and user positions from NED coordinates to Earth-Centered, Earth-Fixed coordinates (XYZ) will be performed, assuming the Earth has an ideal spherical shape by neglecting its oblateness. The transformation of the position coordinates from NED to ECEF consists of three rotations, as shown in Figure 1.

1- Rotation about Y-axis by 90°
2- Rotation about X-axis by θ
3- Rotation about Y-axis by $-\lambda$

These transformations can be expressed in the following matrix form.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -\cos\theta\sin\lambda & -\sin\theta & -\cos\theta\cos\lambda \\ -\sin\theta\sin\lambda & \cos\theta & -\sin\theta\cos\lambda \\ \cos\lambda & 0 & -\sin\lambda \end{bmatrix} \begin{bmatrix} N \\ E \\ D \end{bmatrix} \quad (1)$$

For N = 0, E = 0 and D = -r, the position coordinates in ECEF are

$$\begin{aligned} X &= r\cos\theta\cos\lambda \\ Y &= r\sin\theta\cos\lambda \\ Z &= r\sin\lambda \end{aligned} \quad (2)$$

$$r = Re + h \quad (3)$$

where, Re is the radius of the Earth, h is the altitude of the satellite.

In the opposite way, the position coordinates in NED can be calculated from the position coordinates in XYZ, taking in account the following formulas developed from Figure 1.

$$\sin\theta = \frac{X}{\sqrt{X^2+Y^2}}, \quad \tan\lambda = \frac{Z}{\sqrt{X^2+Y^2}}, \quad r = \sqrt{(X^2+Y^2+Z^2)} \quad (4)$$

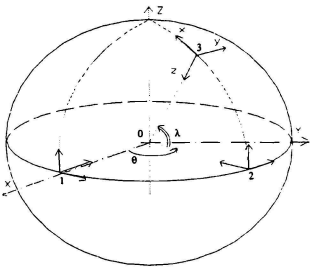


Figure 1. ECEF and NED coordinate systems.

Satellite Navigation System

Navigation in three dimensions is the primary function of GPS, which utilizes the concept of time-of-arrival ranging to determine user position. This concept entails measuring the time it takes for a signal transmitted by an emitter at a known location to reach a user receiver. This time interval, referred to as the signal propagation time, is then multiplied by the speed of the light to obtain the emitter-to-receiver distance. By measuring the propagation time of signals broadcasted from multiple emitters at known locations, the receiver can determine its position. A minimum of four satellites is required to calculate latitude, longitude, altitude, and clock bias error. The satellites also inform the receiver where they are in their orbits above the Earth. If the receiver knows the exact distance from a satellite in space, the receiver knows its location is somewhere on the surface of an imaginary sphere centered at the satellite with radius equal to the

distance to the satellite. If the receiver knows the distance to a second satellite, the receiver must be located somewhere on the circle where the two spheres intersect. If the receiver knows the distance to a third satellite, the receiver lies at either of two points where the three spheres intersect. One of these points is usually unadmissable and the GPS receiver can algorithmically eliminate the inadmissible location.

GPS Measurements

Receiver Position

The mathematical model of how the observations relate to the state of the vehicle is represented by the pseudorange equation.

$$D = C\Delta T \tag{5}$$

where ΔT is the signal travel time estimated by receiver by subtracting the time its clock registers from the time indicated by satellite when it transmitted the relevant pulse. This travel time is multiplied by the speed of light C to obtain the range D to the satellite. Because the clock in receiver is not perfectly synchronized with clock carried onboard the satellite, the receiver actually estimates the false range to each satellite. Thus, the right signal travel time can be written as

$$\Delta T = \Delta T(GPS) + T_0 \tag{6}$$

where T_0 represents the advance of the receiver clock with respect to system time. (Commonly referred to as clock bias error),. Then, equation 5 can be written as

$$D = C[\Delta T(GPS) + T_0) \tag{7}$$

$$D = C\Delta T(GPS) + CT_0 \tag{8}$$

where $C\Delta T(GPS)$ represents the geometric distance between receiver and satellite. The simple model for pseudorange observation between the user and the ith satellite can be related to the user position and clock bias error as following

$$D_i = \|R_i - R_0\| + CT_0 \tag{9}$$

where R_i is the position vector of the ith satellite, R_0 is the position vector of user, i ranges from 1 to 4. Expanding the above equation for each satellite, it will be written as the following set of equations in the unknowns X_0, Y_0, Z_0 and T_0

$$D_i = \sqrt{(X_i - X_0)^2 + (Y_i - Y_0)^2 + (Z_i - Z_0)^2} + CT_0 \tag{10}$$

where, X_1 ,Y_1 and Z_1 denote the ith satellite position coordinates, while X_0 ,Y_0 and Z_0 denote the user position coordinates. Solving four equations for X_0 ,Y_0 ,Z_0 and T_0 , the user's position can be obtained.

These nonlinear equations can be solved for the unknown by using a linearizing approximation. Knowing approximately where the receiver is ($\hat{X}_0$ $\hat{Y}_0$ $\hat{Z}_0$), the offset from true position (X_0 Y_0 Z_0) can be donated by displacements ΔX_0 ΔY_0 ΔZ_0. Expanding with a Taylor series about the approximate position, the position offsets (ΔX_0 ΔY_0 ΔZ_0) can be obtained as linear functions of the known coordinates and peseudorage measurements. These manipulations are described below. Using the approximate position location $\hat{X}_0$ $\hat{Y}_0$ $\hat{Z}_0$ and time bias estimate $\hat{T}_0$, an approximate pseudorange can be calculated as following.

$$\hat{D}_i = \sqrt{(X_i - \hat{X}_0)^2 + (Y_i - \hat{Y}_0)^2 + (Z_i - \hat{Z}_0)^2} + C\hat{T}_0 \tag{11}$$

As stated above, the unknown user position and receiver clock offset are considered to consist of an approximate component and an incremental component:

$$\begin{aligned} X_0 &= \hat{X}_0 + \Delta X_0 \\ Y_0 &= \hat{Y}_0 + \Delta Y_0 \\ Z_0 &= \hat{Z}_0 + \Delta Z_0 \\ T_0 &= \hat{T}_0 + \Delta T_0 \end{aligned} \tag{12}$$

Therefore we can write

$$f(X_0, Y_0, Z_0, T_0) = f(\hat{X}_0 + \Delta X_0, \hat{Y}_0 + \Delta Y_0, Z_0 + \Delta\hat{Z}_0, \hat{T}_0 + \Delta T_0) \tag{13}$$

This function can be expanded about the approximate position and associated predicted receiver clock offsets ($\hat{X}_0$ $\hat{Y}_0$ $\hat{Z}_0$ $\hat{T}_0$) using a Taylor series:

$$f(\hat{X}_0 + \Delta X_0, \hat{Y}_0 + \Delta Y_0, \hat{Z}_0 + \Delta Z_0, \hat{T}_0 + \Delta T_0) = f(\hat{X}_0, \hat{Y}_0, \hat{Z}_0, \hat{T}_0) + \frac{\partial f(\hat{X}_0, \hat{Y}_0, \hat{Z}_0, \hat{T}_0)}{\partial f(\hat{X}_0)} \Delta X_0$$

$$+ \frac{\partial f(\hat{X}_0, \hat{Y}_0, \hat{Z}_0, \hat{T}_0)}{\partial f(\hat{Y}_0)} \Delta Y_0 + \frac{\partial f(\hat{X}_0, \hat{Y}_0, \hat{Z}_0, \hat{T}_0)}{\partial f(\hat{Z}_0)} \Delta Z_0 + \frac{\partial f(\hat{X}_0, \hat{Y}_0, \hat{Z}_0, \hat{T}_0)}{\partial f(\hat{T}_0)} \Delta T_0 + \ldots\ldots$$

The expansion will be truncated beyond the first-order partial derivatives to eliminate nonlinear terms. The partial derivatives evaluate as follows:

$$\begin{aligned} \frac{\partial f(\hat{X}_0, \hat{Y}_0, \hat{Z}_0, \hat{T}_0)}{\partial f(\hat{X}_0)} &= -\frac{X_i - \hat{X}_0}{\hat{d}_i} \\ \frac{\partial f(\hat{X}_0, \hat{Y}_0, \hat{Z}_0, \hat{T}_0)}{\partial f(\hat{Y}_0)} &= -\frac{Y_i - \hat{Y}_0}{\hat{d}_i} \\ \frac{\partial f(\hat{X}_0, \hat{Y}_0, \hat{Z}_0, \hat{T}_0)}{\partial f(\hat{Z}_0)} &= -\frac{Z_i - \hat{Z}_0}{\hat{d}_i} \\ \frac{\partial f(\hat{X}_0, \hat{Y}_0, \hat{Z}_0, \hat{T}_0)}{\partial f(\hat{T}_0)} &= -C \end{aligned} \tag{15}$$

where $\hat{d}_i = \sqrt{(X_i - \hat{X}_0)^2 + (Y_i - \hat{Y}_0)^2 + (Z_i - \hat{Z}_0)^2}$

Substituting Equation 11 and Equation 15 in Equation 14 gives the following equation

$$D_i = \hat{D}_i - \frac{X_i - \hat{X}_0}{\hat{d}_i}\Delta X_0 - \frac{Y_i - \hat{Y}_0}{\hat{d}_i}\Delta Y_0 - \frac{Z_i - \hat{Z}_0}{\hat{d}_i}\Delta Z_0 - C\Delta T_0 \tag{16}$$

After completing the linearization with respect to the unknowns, the above expression will be rearranged with the known quantities on the left and the unknown quantities on the right, yielding

$$D_i - \hat{D}_i = -\frac{X_i - \hat{X}_0}{\hat{d}_i}\Delta X_0 - \frac{Y_i - \hat{Y}_0}{\hat{d}_i}\Delta Y_0 - \frac{Z_i - \hat{Z}_0}{\hat{d}_i}\Delta Z_0 - C\Delta T_0 \tag{17}$$

Simplifying the above equation by introducing new variables

$$\begin{aligned} \Delta D_i &= D_i - \hat{D}_i \\ a_{xi} &= -\frac{X_i - \hat{X}_0}{\hat{d}_i} \\ a_{yi} &= -\frac{Y_i - \hat{Y}_0}{\hat{d}_i} \\ a_{zi} &= -\frac{Z_i - \hat{Z}_0}{\hat{d}_i} \end{aligned} \tag{18}$$

where these terms denote the direction cosines of the unit vector pointing from the approximate user position to the ith satellite in the ECEF frame. For the ith satellite, this unit vector is defined as

$$a_i = (a_{xi}, a_{yi}, a_{zi}) \tag{19}$$

Then, it can be written as

$$\Delta D_i = a_{xi}\Delta X_0 + a_{yi}\Delta Y_0 + a_{zi}\Delta Z_0 - C\Delta T_0 \tag{20}$$

Making ranging measurements to four satellites can solve the four unknowns. The unknown quantities can be determined by solving the set of linear equations below:

$$\Delta D_i = a_{xi}\Delta X_0 + a_{yi}\Delta Y_0 + a_{zi}\Delta Z_0 - C\Delta T_0 \tag{21}$$

These equations can be written in matrix form

$$\Delta D = A\Delta X \tag{22}$$

which has the solution

$$\Delta X = A^{-1}\Delta D \tag{23}$$

Where

$$\Delta D = \begin{bmatrix} \Delta D_1 \\ \Delta D_2 \\ \Delta D_3 \\ \Delta D_4 \end{bmatrix} A = \begin{bmatrix} a_{x1} & a_{y1} & a_{z1} & -1 \\ a_{x2} & a_{y2} & a_{z2} & -1 \\ a_{x3} & a_{y3} & a_{z3} & -1 \\ a_{x4} & a_{y4} & a_{z4} & -1 \end{bmatrix} \Delta X = \begin{bmatrix} \Delta X_0 \\ \Delta Y_0 \\ \Delta Z_0 \\ C\Delta T_0 \end{bmatrix} \tag{24}$$

Once the unknowns are computed, the user's coordinates (X_0 Y_0 Z_0) and receiver clock offset T_0 are then calculated using Equation 12.

Geometric dilution of Precision

The geometry of the visible satellites is an important factor in achieving high quality results, especially for point positioning. This geometry changes with the time due to the relative motion of the satellites. GDOP is computed from the geometric relationships between the receiver position and the positions of the satellites. Geometric Dilution of Precision can be calculated from the inverse of a normal matrix formed from A defined in Equation 24. The cofactor matrix Q is calculated as following

$$Q = (A^T A)^{-1} \tag{25}$$

where

$$Q = \begin{bmatrix} q_{xx} & q_{xy} & q_{xz} & q_{xt} \\ q_{xy} & q_{yy} & q_{yz} & q_{yt} \\ q_{xz} & q_{yz} & q_{zz} & q_{zt} \\ q_{xt} & q_{yt} & q_{zt} & q_{tt} \end{bmatrix}$$

Considering all the sources of error. This optimum value might occurs when one satellite is directly overhead and the others are close to the horizon, spaced in a circle, separated from each other by 120 degrees. This description means that a good GDOP can be obtained when a volume formed by the satellites is large and the elevation angles from receiver to the set of satellites are different. On the other hand, a poor GDOP can be obtained when a volume formed by the satellites is small and the elevation angles from receiver to the set of satellites are similar.

In general, ranging errors from the satellite signals are multiplied by the appropriate GDOP term to estimate the resulting position or time error. Various DOP terms that are in common use are useful to characterize the accuracy of various components of the position and time solution. These terms are introduced as follows.

GDOP = Geometric Dilution of Precision.

$$GDOP = \sqrt{q_{xx} + q_{yy} + q_{zz} + q_{tt}} \tag{26}$$

GDOP = Position Dilution of Precision (3-D), sometimes the Spherical DOP.

$$PDOP = \sqrt{q_{xx} + q_{yy} + q_{zz}} \tag{27}$$

HDOP = Horizontal Dilution of Precision (Latitude, Longitude).

$$HDOP = \sqrt{q_{xx} + q_{yy}} \tag{28}$$

VDOP = Vertical Dilution of Precision (Height).

$$VDOP = \sqrt{q_{zz}} \tag{29}$$

TDOP = Time Dilution of Precision (Time).

$$TDOP = \sqrt{q_{tt}} \tag{30}$$

Although, each of GDOP terms can be individually computed, they are not independent of each other, because they are formed from the covariance matrix Q. A high TDOP (time dilution of precision), for example, will cause receiver clock errors that will eventually result in increased position errors.

Test Case

Mathematically, the receiver requires the distances from three satellites and their coordinates X_i, Y_i and Z_i to compute the navigation fix. Measurements collected simultaneously from three satellites are processed to solve for the three components of position. The fourth satellite measurement is needed to solve for the unknown clock error ΔT_0.

For example, suppose that the optimal group of four visible GPS satellites are described by their positions in ECEF coordinates as follows (all dimensions in Km.).

x1 = 18476,	y1 = 8811.8,	z1 = 1500.3
x2=9938.5,	y2=1371,	z2=-10923
x3=17523,	y3=-9414.9,	z3=-3101.2
x4=6726.7,	y4=-15104,	z4 = 11362

Considering the position of the user in the same Coordinate system is given as below

xo = 6357.13, yo = 0, zo = 0

According to the equations 24 and 25, matrices A and Q will be formed respectively as following.

$$A=\begin{bmatrix} -0.8048 & -0.5852 & -0.0996 & -1.0000 \\ -0.2001 & -0.7664 & 0.6104 & -1.0000 \\ -0.7478 & 0.6306 & 0.2077 & -1.0000 \\ -0.0196 & 0.7990 & -0.6010 & -1.0000 \end{bmatrix}$$

$$Q=\begin{bmatrix} 2.2837 & -0.0783 & 0.3131 & -1.0042 \\ -0.0783 & 0.8191 & 0.7823 & 0.0736 \\ 0.3131 & 0.7823 & 2.0896 & -0.0621 \\ -1.0042 & 0.0736 & -0.0621 & 0.6945 \end{bmatrix}$$

Then, the value of GDOP factor is 2.4264.

Code Structure

Programming code is written using MATLAB software V. 5.2. The object of this program is to calculate the GDOP factor. This program consists of the following steps:

-Input the position data of the satellites (Latitude, Longitude, Altitude) taken from the numerical simulation of GPS satellites using the STK software.

-Transform the positions of both the observer and satellites from the NED coordinate system to the ECEF coordinate system.

-Determine the number of visible satellites, and form a combination of groups of four satellites.

-Calculate GDOP values for each group, find the optimal one that gives minimum GDOP value, specify the satellites of the optimal group and plot the results.

Observable Satellite

Operational satellite geodesy is based upon data transmitted from the satellite to the user by means of electromagnetic waves generated by an oscillating electric force. Because the satellites are so far away, their radio signals are weak. Therefore, the GPS receiver to locate a satellite, the satellite must be above the horizon and unobstructed by building, mountains, etc. At any given moment at any point on the planet, there are between six and nine satellite above the horizon. The satellite will be in view if the user can receive the signal from the satellite. To simulate this physical behavior in the programming code to know whether the satellite is either in view or not. A satellite will be considered in view if the distance D between the satellite and user is less than the distance d between user and satellite, when the satellite has zero elevation angle, as shown in Figure 2. The distance d is calculated from the following formula.

$$d = \sqrt{(r)^2 - (\mathrm{Re})^2} \qquad (31)$$

While the distance D is calculated from Equation 9.

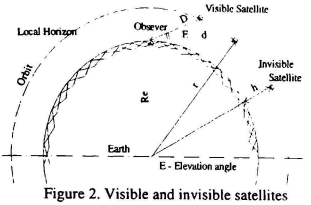


Figure 2. Visible and invisible satellites

Observable Groups

At every time there is a different number of satellites in view with respect to a specified observer position. The GDOP value will depend on which group of four satellites the user selects to determine its position. To find the group that gives the optimal value of GDOP, GDOP must be calculated for all groups formed from the satellites in view. The number of groups can be determined by using the following combination formula

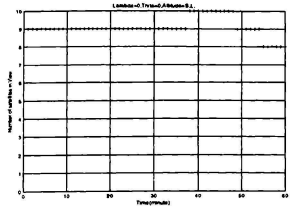
$$Number\ \ of\ \ groups = \frac{\prod_{i=0}^{m-1}(n-i)}{m!} \qquad (32)$$

Where, n is the number of visible satellites, m is the number of satellites in each group. For example, if there are 6 satellites in view, the number of groups containing four satellites is 15. The members of each group for a variable number of viewable satellites are generated by a Matlab subroutine developed by the author.

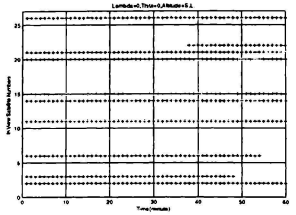
Examples of Application

Different observer's positions will be tested to illustrate how the user position affects GDOP value. Observer's position, number of satellite in view, in view satellite numbers, optimal in view satellite numbers and GDOP factor will be calculated. Simulation time will be 60 minutes to see how the geometry of visible satellites affects the GDOP value. The results are shown in the following figures 3, 4, and 5.

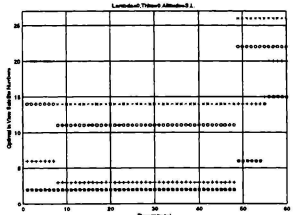
From these figures, it is clear that GDOP factor and the number of visible satellites change with the time for the same user's position. Continuos GDOP factor occurs with the same group of satellites in view which gives us the minimum value of GDOP, only the geometric positions of these satellites change while, the discontinuity in the GDOP factor reflects the situation when one or more of visible satellites move in or move out of optimal group.



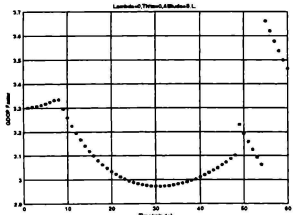
3-a



3-b

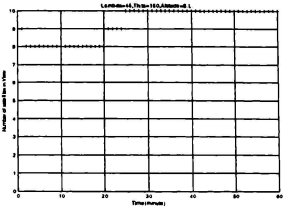


3-c

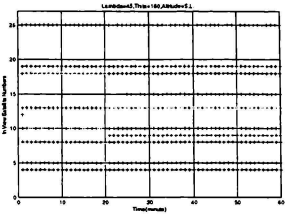


3-d

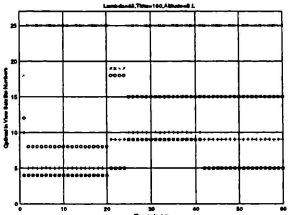
Figure 3. Observer's position at θ=00, λ=00 and h=0



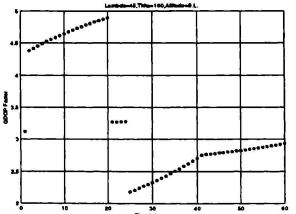
4-a



4-b

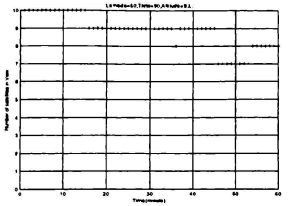


4-c

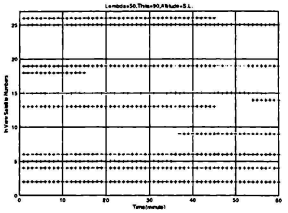


4-d

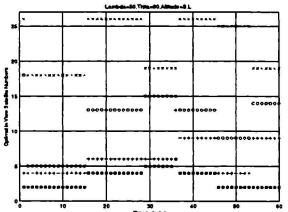
Figure 4. Observer's position at θ=160, λ=45 and h=0



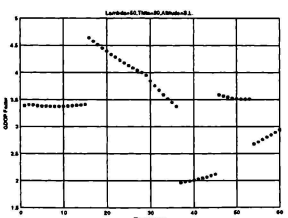
5 - a



5 - b



5 - c



5 – d

Figure 3. Observer's position at θ=90, λ=50 and h=0

Conclusions

In this research, the main concepts of GPS are studied and analyzed, powerful software tools is used to extract the required data of GPS satellite positions to implement in navigation algorithms. This algorithms is developed using MATLAB software tool box to determine in view satellites and to calculate GDOP factor values relative to different observer's position during specified time period

Acknowledgment

This research was done as a part of research skill required Ph.D. Degree under supervision of Dr. Brett Newman. His help is appreciated.

References

1. Analytical Graphics, " Satellite Tool Kits, IN VIEW," 1998.

2. B.Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, GPS Theory and Practice, Springer – Verlag Wien, New York, 1997.

3. Bradford W. Parkinson and James J. Spilker Jr., "Global Positioning System: Theory and Applications", Volume I and II, Progress in Astronautics and Aeronautics, Volume 163, Washington, DC, 1996.

4. Brian L.Stevens and Frank L. Lewis, Aircraft Control and Simulation, John Wiley & Sons, 1992.

5. Elliott D. Kaplan, Understanding GPS Principle and Applications, Artech House, Norwood, MA, 1996.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37243

AIAA-2000-4185

HYBRID-SYSTEM SIMULATION FOR NATIONAL AIRSPACE SYSTEM SAFETY ANALYSIS

A.R. Pritchett*, S.M. Lee†, D. Huang†, and D. Goldsman‡
Georgia Institute of Technology

ABSTRACT

Analysis of large, complex systems requires simulations of hybrid-system dynamics, i.e. dynamics which are best described by a combination of continuous-time and discrete-event models, and their interactions. To serve as a valuable research tools, such simulations need also be computationally efficient, readily-modifiable, and open to a wide-range of component modules. This paper describes the development of a simulation architecture meeting these criteria. The issues with its development are described conceptually, and its application to the task of safety analysis of the national airspace system is discussed. In particular, an object-oriented approach to hybrid-system simulation is detailed, and computationally efficient methods of updating the simulation are described and compared.

INTRODUCTION

The behavior of many large, complex systems is hybrid in nature; i.e. it includes both continuous-time and discrete-event models, and the behaviors these models represent are not separable, but instead can interact in significant ways. A simulation capable of recreating these hybrid-system dynamics provides an analysis tool that can dramatically change the design process. In electronics, for example, integrated circuits were purposefully designed to have components spaced far apart until simulations capable of predicting electro-magnetic interference could be used to analyze and re-design smaller chips[1]. Many aerospace systems also are best captured best by hybrid-system simulations, ranging from aircraft with flight control systems that change modes, to on-board systems with discontinuous behaviors such as open-closed mechanisms.

Take, for example, the task of performing safety analysis on the National Airspace System (NAS). Merely simulating the trajectories of the aircraft would not capture the discrete actions of controllers; likewise, continuous time simulation architectures would not be well suited for the tasks of inserting aircraft into the airspace being simulated at random rates, or for the injection of stochastic injection of disturbances. Elements of NAS dynamics have been simulated using purely discrete-event models[2]; however, such models do not have the resolution to capture safety issues. Likewise, hybrid-system simulations have been made of the NAS, but have been limited to specific applications or parts of the NAS[2,3].

In order to serve as an effective design tool, simulation of large-scale systems must also meet a number of practical considerations. Most obviously, the simulation should be sufficiently computationally efficient that simulation can provide a time-effective analysis tool, even when large number of runs are required. In addition, the simulation should be rapidly reconfigurable, so that the development time of the simulation is not prohibitive, and so that the simulation can be applied to a range of applications and can accommodate models of varying fidelity.

This paper discusses issues with hybrid-simulation with the thesis that many of these issues can be solved by an object-oriented software architecture. This architecture handles the communication between objects without needing to treat objects differently based on the type of their underlying model. Likewise, this architecture can be constructed to control the timing and updating of the elements in a computationally efficient manner.

First, a comparison of fundamental differences and similarities between continuous-time and discrete-event models is made. Then, the test case used in this paper – the safety analysis of the NAS – is described. Conceptual issues in, and requires of, hybrid-system simulations are discussed, and then an example simulation software architecture is described. Methods of controlling simulator timing are described, illustrated by comparisons from the test case.

* Assistant Professor, Schools of Industrial and Systems Engineering and Aerospace Engineering, Member AIAA
† Graduate Research Assistant
‡ Professor, School of Industrial and Systems Engineering
Copyright © 2000 The American Institute of Aeronautics and Astronautics Inc. All rights reserved.

BACKGROUND: DISCRETE EVENT AND CONTINUOUS TIME MODELS

Important, fundamental differences exist between discrete event and continuous-time models. These differences are shown in Table 1.

Discrete event models typically attempt to define the state of a system by categorizing whether conditions exist, or by quantifying the number of entities within a category. As such, they can describe the system without attempting to explain internal dynamics. Their defining parameters stipulate how and how often states will transition from one to another. These parameters are set by experimental observation of existing systems, and can capture the stochastic nature of the system. As such, discrete-events are well-suited for modeling systems made of multiple entities with no internal dynamics of relevance – and discrete-event models typically require study of an established system to ascertain their parameters.

Continuous-time models, on the other hand, typically attempt to model the internal dynamics within the system. The state is typically defined as a measure of energy within the system, such as measures of position and velocity. These models are often physics-based; i.e. they can be developed before the system can be measured as a form of analysis. However, these models are often described by differential equations, which can be numerically complex to propagate forward through time. (A related model type, that of discrete-time systems, is modeled in similar ways such as the use of difference equations[4], and can treated as a special class of continuous-time models).

These two types of models, applied to the same problem, tend to have very different rates at which the states need to be updated. Continuous time models are solved through numerical integration (or transition) algorithms which approximate the continuous variations by updates at discrete intervals. These intervals (or timesteps), at the very least, must be at twice the rate that dynamics of interest occur in order to capture their basic properties[5] - but in most applications, the timestep is set much smaller to reduce error in the numerical solutions[6]. Discrete events, on the other hand, typically capture fairly large changes in dynamics, and need to occur less often. In some cases, the update rates for the two types of models may be separated by several orders of magnitude.

Comparisons of these two types of models are conceptual distinctions only. It has been proven possible to incorporate models of either type into simulation software intended for the other. For example, continuous time models have been fit into discrete event simulations by fitting updates in their state values into mechanisms for discrete transitions with fixed, small transition times. However, it has also been noted that such cross-implementation often requires restrictive assumptions on the models, limits their accuracy, increases the complexity of the software, and often doesn't result in a computationally efficient simulation.[7] As such, these differences are of dramatic import to the simulation designer, as the simulation architecture is typically tailored to the type of model implicit in the simulation.

Table 1. Comparison of Continuous-Time and Discrete-Event Models

	Continuous-Time Models	Discrete-Event Models
System Being Simulated	Specific mechanical unit with complex functioning	Multiple, often simple, entities
Definition of System State	Distribution of energy within system	Categorization of current system properties
Typical Measures of System State	Position, attitude, velocity (Deterministic)	Queue size, Incidence (Statistical properties thereof)
Typical Factor Driving Updates	Time	Transitions from state to state
Capability of Simulation	Analyze deterministic dynamic behavior of a mechanical unit	Analyze stochastic nature of interactions between entities
Common Uses	Design and analysis of unit, (real-time) training	Analysis and planning of operations with multiple entities

TEST CASE: SAFETY ANALYSIS OF THE NATIONAL AIRSPACE SYSTEM

At its full extent, the National Airspace System (NAS) is a system of overwhelming complexity. Thousands of aircraft may be aloft at one time; hundreds of controllers are monitoring and directing them with the assistance of many communication and surveillance technologies.

Complexity of the NAS is also due to the large number of different entities involved in its operation. Only at a high-level can the NAS be modeled as consisting of controllers, pilots and aircraft. When more resolution is required, distinctions must be made between the different types of controllers (ground, tower, terminal area, en-route, etc.) and their hierarchies; likewise, aircraft can be of many different types with different performance, and their pilots may have widely different goals and levels of experience.

The behavior of the NAS, to a great extent, is defined by the interaction of these different elements. The NAS can not be modeled as a collection of independent aircraft flying simultaneously; instead, controllers (and pilots) are constantly changing direction of flight in response to the actions of others. These interactions may meet a number of goals, ranging from time-critical collision avoidance maneuvers, to strategic plans for air traffic flow management.

In order to meet increasing capacity demands and stricter safety demands, the NAS is being updated. These updates range in scale from near-term equipment upgrades, to longer-term calls to change the manner in which controllers and pilots interact.

These upgrades create a design problem of vast scale. For both cost and safety, as much analysis should occur before implementation as possible. The worth of this analysis will be measured by its ability to predict problems, allowing for their correction, before implementation. Simulation and modeling have been recognized as a critical part of this analysis[8].

The NAS has been simulated before[2,3]. However, most of these simulations have not been suitable for large-scale safety analysis. Many NAS simulations have been motivated by studies of efficiency. As such, these simulations have been concerned with values such as airport inter-arrival times, or flight delays. These concerns are best abstracted by discrete events, and hence these have largely been covered by discrete event simulations.

Safety, on the other hand, is largely determined by continuously-evolving interactions which discrete event simulations can not capture. For example, a discrete event simulation may model when two aircraft arrive at an airport, but without monitoring their continuous trajectory, it is near-impossible to measure with certainty whether these two aircraft came unacceptably close during their flights. Therefore, simulation suitable for safety analysis needs better resolution of some parts of the system than can be readily provided by discrete events alone.

The most notable element requiring adequate resolution is the trajectory of the aircraft itself. Typically represented by differential equations, aircraft trajectories are usually best modeled as continuous time objects. Fortunately, many excellent models are currently available at all levels of fidelity, ranging from outer-loop models of the aircraft's guidance, to detailed inner-loop models of the aircraft's flight dynamics[9,10,11]. These models are physics-based, which allows their parameters to be estimated (if not exactly known), and brings predictive power to the simulation where novel NAS changes are to be simulated before measurements of actual dynamics can be observed.

Other elements remain best modeled as discrete events, for a variety of reasons. Some elements of NAS behavior can be predicted to occur discretely; for example, the generation of a cue of aircraft waiting for taxi clearance to the take-off runway is discrete in nature. Other elements of the NAS are not needed at a fine resolution, and so computational effort can be saved by reducing them to a notable state – for example, a detailed, computationally-expensive continuous time model of an aircraft waiting for take-off can be temporarily replaced by a notation in the take-off cue. Finally, discrete events can be used to interject stochastic elements into the simulation, including the dramatic conditions caused by errors and failures, as well as disturbances into aircraft model parameters representing variations in aircraft behavior.

Of critical importance in NAS safety simulation is inclusion of human performance models, for their behavior drives system dynamics. Their behavior should not be typecast into either continuous or discrete forms. For example, many human performance models can be based on procedures or expert systems that call for isolated actions to be triggered by conditions in the environment (discretely) while also maintaining a continuously evolving valuation of workload or working memory content.

Measurements of NAS safety can also be treated as discrete events that occur when conditionals in the environment are met, such as loss of separation between two aircraft. Unlike 'normal' discrete events, these measurements do not need to trigger subsequent events in the simulation, beyond recording the event to an output file. However, these measurements can share in the computational structures that are used for other discrete events.

HYBRID-SYSTEM SIMULATION CONCEPTS

In cases where an elegant, analytic solution can not be found to analyze a system represented by discrete event and continuous time models, a hybrid-system simulation is required to calculate system dynamics through time. Several approaches have been suggested to the design of hybrid-system simulations. Some create larger 'meta-model' frameworks in which each 'type' of simulation functions separately[1..12]. Other solutions have adapted existing simulations of one type to include the other[13,14].

A third approach, sometimes called the 'fully-integrated approach'[1], seeks to create new software that inherently accepts the two model types. Such approaches have been undertaken with special modeling languages[15]. This paper will instead focus on software architectures (not necessarily written in special language) that can accept models of any type, control their timing in computationally efficient mechanisms, and handle communication between the objects.

Previous sections have highlighted the differences between the models used in hybrid-system simulation. A simulation architecture can capitalize upon what the abilities these models share: to update themselves when required; to report when their next update is required; and to report interactions with other objects that warrant a joint update.

As such, at a high-level, a simulation architecture can require components to meet these three interface requirements. All other dynamics of the components can remain internal to their models, without requiring intervention by the larger simulation architecture. This internalism can be considered a feature. It prevents fundamental restrictions on the type of model allowed in the simulation – and as such it allows for the simulation to include components of various resolutions as required by the task at hand[16,17].

Without placing restrictions on components' models, the simulation architecture also needs to support their interactions. These interactions may take on several forms. Traditional to continuous time simulation is coupling between different continuous time objects, such as two aircraft flying in formation (or executing avoidance maneuvers) reacting to the movements of each other. A discrete event may also impact a continuous time object in several ways: it may enact a discrete change in the variables maintained by the continuous time object (such as a sudden change in acceleration due to the application of brakes or engine failure); it may change a parameter's value within the continuous time object (such as a change in stability derivatives in response to a discrete change in aircraft configuration); or it may 'swap in' a whole new continuous time model better suited to the situation (such as inclusion of a higher fidelity aircraft model at the start of an avoidance maneuver, or a switch to a 'taxiing' aircraft model after landing). Continuous time objects can interact with discrete events when their values are the events' triggers.

Within a simulation framework that supports such interactions, the simulation designer is given the ability to make components work efficiently on their own. For example, in modeling an aircraft with on-board systems that have discrete dynamics, the simulation designer has choices beyond the common requirement that these two behaviors be kept common in one model, despite their two different time-scales. Instead, the designer would have the option of making the on-board systems into a separate discrete-event model that communicates appropriately with the continuous-time model of the aircraft dynamics. Such an approach allows the simulation designer to separate behaviors according to the times at which they will need to be updated without substantial re-workings within individual components[15].

While a simulation's operation does not depend on measurements, the desire for accurate measurements are typically the motivation for the simulation. In many cases, the measurements are temporal in nature and hence it is important that the measurement be taken exactly when it occurs. One approach is to make measurements an active element by treating them as discrete events that must report when they must next be updated. This projected update time can be a conservative estimate of when the conditions wanting recording may next occur. While this process requires measurements to have a prediction power, it also reduces the need for unnecessary measurements. This also mirrors a duality between measurements and conditionally-based discrete events – while the former have no lasting impact on simulation dynamics, the latter are measurements with a consequence.

Based on this discussion, several requirements for a hybrid-system simulation architecture may be summarized. First, the simulation architecture should not place unnecessary limits on the types of objects, but instead accommodate any components that can list their update times and update themselves upon command. Second, the simulation architecture should facilitate communication between objects, so that it is easy for the simulation designer to break apart models according to their functionality and update rates, without requiring lengthy communication standards to be developed. Finally, the simulation architecture should be capable of timing the updates of the individual components in a computationally efficient manner.

SIMULATION ARCHITECTURE

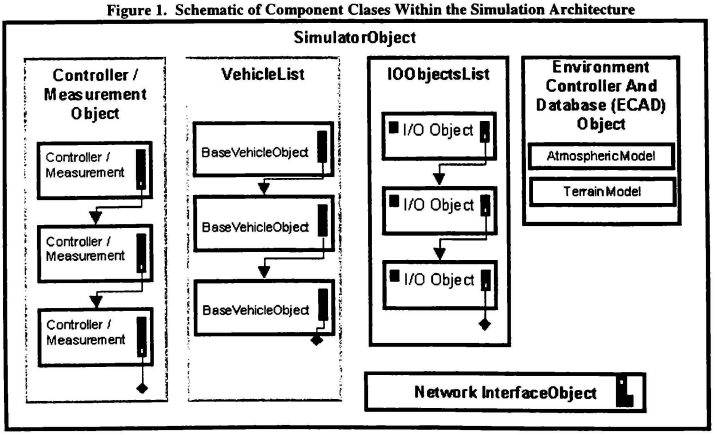
The previous sections discussed conceptual issues with hybrid-system simulation. This section discusses a particular simulator architecture design. This simulation is based on extensions to and adaptations of the Reconfigurable Flight Simulator (RFS), which was originally designed for more traditionally applications of flight simulation.[18]

This simulation allows for the inclusion of several broad classes of objects, as shown schematically in Figure 1. An arbitrary number of vehicles can be added to the vehicle list. These components must fit a base interface for vehicles, which was designed to support continuous-time models of vehicle dynamics. For specific applications, base interfaces have also been defined for aircraft and for spacecraft, and several basic components, including 6DOF and waypoint following aircraft have been developed which can be used or extended by other developers.

Beyond these capabilities as a 'normal' flight simulator, an arbitrary number of controller and measurement objects can be added to a dedicated list. The base interface standards for these objects is less specified, allowing flexibility in the objects' behavior. Components that have been added to date include 'Random Aircraft Generators' that place aircraft into airspace according to a given distribution of inter-arrival times, basic air traffic controllers that determine aircraft sequences in merging arrival streams and then command speeds to aircraft to maintain proper spacing within the traffic streams, and measurement objects which look for and record events specified in a text script. The flexibility of this type of component allows for many other types of discrete event and measurement components to be added as desired.

Other types of components are also available in the simulation architecture as support for hybrid-system simulation needs. Most of the components needed for this application are already established, but these components can be modified or added to as desired. Input-output (IO) objects provide mechanisms for graphical output of the simulation (such as an 'air traffic control screen' and text output of commands given by controllers), and for data recording of the measurements. The environment controller and database (ECAD) provides a common simulation environment for all the components by providing common axis definitions and conversions, and by allowing for the inclusion of atmospheric and terrain models as desired. Finally, a networking object handles communications between simulations run on multiple machines over a network; this networking is transparent to the rest of the simulation and therefore does not require re-compilation of components to use or access networking.

Figure 1. Schematic of Component Clases Within the Simulation Architecture



To function as a hybrid-system simulation, each continuous time, discrete event and measurement object is required to meet the minimal standards of a standard interface: i.e. each of these components must update their state upon command, report the time of their next update (or, in the case of some discrete event and measurement objects, the next time at which an update might occur conditionally upon other events), and must identify whether any other objects must also be updated.

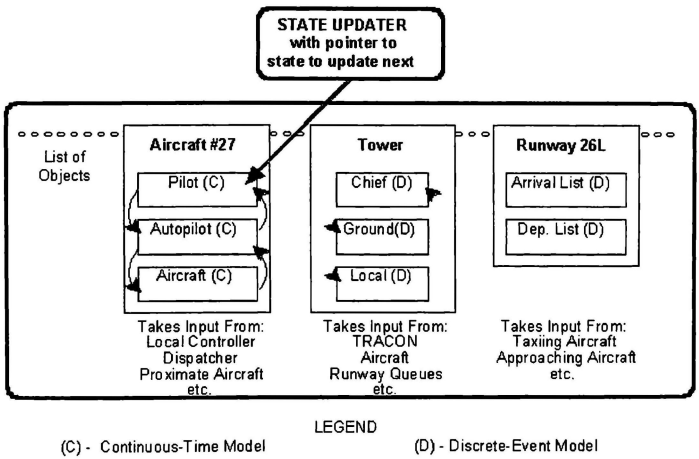
For the continuous time objects, a standard input that needs to be provided is an upper bound on the numerical error allowable in each timestep. With this input, the next update time is calculated based upon knowledge of their interior dynamics ; algorithms for such flexible time step calculations are well established[5].

Communications between objects is handled by the high-level simulation object. As such, the designer of a hybrid-system simulation using this architecture does not need to develop communication standards beyond giving individual components the ability to send and receive messages. Many standard messages can be passed through the base interface standards of vehicle and controller / measurement objects.

Those messages that don't fit within the base interface standards can be sent through simulator object via the Object Data / Methods Extensions (OD/ME) protocol, which requires sending a message to the simulator object along with a request for its destination[18]. This mechanism also allows for objects to request the addition or destruction of other objects; for example, a random aircraft generator can request the addition of a new aircraft to the vehicle list, with subsequent messages to that new aircraft that provide it with initialization data.

To facilitate efficient timing of the objects' updates, a 'state updater' object within the controller list maintains a list of the vehicle and controller objects, sorted by the time of their next desired update. As such, this state updater object can identify the object next to be updated, regardless of type, can query that object as to whether it requires other objects to also be updated, and can command the appropriate objects to update. Once objects have been updated, they each are asked for their new update time, and are sorted accordingly.

Figure 2. Schematic of Sorting of Simulation Components by Update Time, With Access by a State Updater Component



EFFICIENCY AND TIMING METHODS

In large-scale simulation, concerns with computational efficiency extend past standard efforts to make each component individually efficient. Overall efficiency is achieved when each object updates only when needed to meet several criteria: accurate modeling of its interior dynamics; correct interaction with other objects; and timely measurements.

Any unnecessary updates of objects may be considered wasted use of the processor. However, methods of deciding when an update may be required for correct interactions or measurements is usually non-exact once the simulation at hand is non-trivial in size. Likewise, the overhead computation to enable the most sophisticated timing methods may be non-trivial and can slow down the simulation by themselves.

Even measuring the computational efficiency of a simulation can be non-trivial. Once the simulation includes stochastic elements, it can be difficult to compare with certainty the relative speed of different update timing methods, who may take the simulation though different system dynamics due to the inclusion of disturbances or anomalies.

This section will compare different timing methods, and illustrate their effect on a representative NAS simulation using the architecture described in the previous section. Then trade-offs in the fundamental characteristics of these methods are discussed, and alternative methods are commented on.

Timing Methods

Two major variables define the variety of methods for determining the timing of simulation components: first, there is a selection of how the time step is set (next-event time advance, or fixed-increment time advance)[19]; and second is selection as to whether the simulation will be entirely synchronous (i.e. all components update at the same time), partially synchronous and partially asynchronous; or entirely asynchronous (i.e. all components update individually.)

Several timing methods can be defined by these two variables, as follows.

Fixed Time-Step Synchronous This timing method has all objects update at the same time, and this update time is based upon a fixed time step. This method is commonly used in current flight simulation techniques, where the time-step may be fixed by conservative analysis of the fastest dynamics in the system, or by the system clock in real-time simulation. This method is very basic and is often the first step in development of a hybrid-system simulation. It also provides conservative results that can be guaranteed to not miss any measurements or interactions by the setting of an arbitrarily small time step, without requiring predictions from discrete event or measurement objects. However, it also forces all objects to update at a rate governed by the worst-case dynamics of the component with the fastest response, which is computationally inefficient

Variable Time-S[illegible] Synchronous This timing method has all the objects update at the same time, but varies the update time from one timestep to the next to meet the needs of the simulation. For instance, the update time may be chosen by polling all objects for their desired time step, and then selecting the worst-case (smallest). This method still forces some objects to update more times than normally required, but can relax the time step when conditions allow.

Asynchronous With Resynchronization This timing method allows for components to be updated independently following their own update times. This is shown schematically in Figure 3 for a simulation with four aircraft, a random aircraft generator (RAG) and a measurement object; the aircraft and RAG update at their own rates until a measurement is requires a complete synchronization. This will allow for objects with fast dynamics to update frequently without requiring other objects to be bound by such small time steps. However, this method also allows for objects which interact, or which measure interactions, to require all objects (or some objects) to resynchronize when it is time for their update, with the result that interactions and measurements can be based on values from temporally co-located objects.

Case Study: Simulation of a Standard Terminal Arrival Route

To demonstrate the computational efficiency of these methods, a numerical simulation was conducted using the simulation architecture described in the previous section. The simulation modeled the stream of arriving aircraft flying the Macey Two Standard Terminal Arrival Route (STAR) into Atlanta-Hartsfield airport. Aircraft were injected into the simulation stochastically with a specified inter-arrival rate. A controller scheduled the aircraft from the multiple entry streams into one arrival flow by selecting the appropriate order of the aircraft, and commanding speeds to the aircraft that created this desired traffic pattern. The aircraft were removed from the simulation when they reached the runway.

The results of this simulation are shown in Figure 4. Efficiency is measured by the average number of times the aircraft objects are called to update during the course of the run because it 'counts' the number of updates of the most computationally intensive object – the aircraft – due to both its own desired rate as well as forced resynchronizations.

Figure 3. Schematic of Asynchronous Simulation With Resynchronization

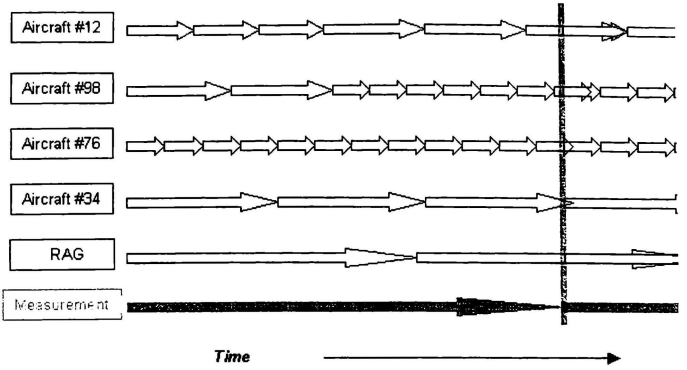
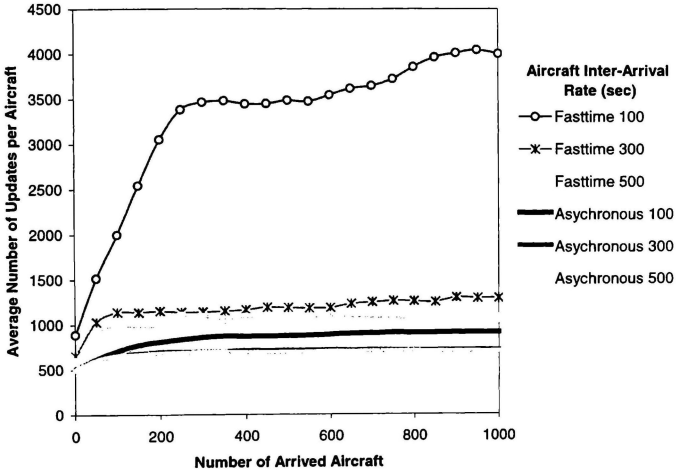


Figure 4. Experimental Results From Runs of a Simulation of a Standard Terminal Arrival Route Showing Computational Efficiency (Measured by Average Number of Calls to the Aircraft) for Synchronous, Variable Time Step and for Asynchronous with Resychnronization Timing Mechanisms



The data is shown for two timing methods. The 'Fast Time' method used the variable time step synchronous method. The 'Asynchronous' method used the asynchronous with resychronization method; the controller and measurement objects commanded a complete resynchronization at times when they predicted a conflict might occur or the next command might be warranted.

The inter-arrival rate of aircraft into the arrival route was also varied. The highest inter-arrival rate (500 seconds) created a fairly low traffic density, with commensurately few interactions. At this inter-arrival rate, the benefits of asynchronous simulation are noticeable, but not dramatic.

The lowest inter-arrival rate (100 seconds) created a high traffic density, in which controller commands and potential conflicts were often possible. The aircraft often needed to maneuver, and conflicts were often possible. In the 'Fast-Time' method, this had the effect of requiring many more updates for all aircraft on average. In the 'Asynchronous' method, fewer updates were required overall as the aircraft needing updates at small intervals were able to work independently.

Trade-Offs Between Resynchronization Intervals and Efficiency

In applications such as just shown in the case study, there appears to be benefits with allowing simulation objects to run, at least for intervals, asynchronously. At first glance, this appears to imply that the best efficiency will arise with the largest resynchronization intervals. However, two main issues limit the size of resynchronization intervals.

First, larger resynchronization intervals require better (and more computationally expensive) predictions by the individual components about when a resynchronization may occur. Better predictions require more extensive calculations – at an extreme, the predictor would need to internally simulate other objects in order to accurately predict when a problem might occur! As such, the value of better predictions can reach a point of diminishing returns, where the additional computations in the predictions offset any savings in computations by other objects.

Second, larger resynchronization intervals require better (and model specific) predictions by the individual components about when a resynchronization may occur. Simple predictions about a potential aircraft collision, for example, can be made based on commonly available aircraft position and velocity; more accurate predictions require knowledge the aircraft's internal dynamics and likely future actions. This imposes an obvious development cost on the simulation. It also makes such 'smart' predictors difficult to use in simulations where a large variety of objects may be involved in the prediction, and limits the use of the predictors to specific cases.

Alternatives to Resynchronization

So far, this discussion on simulation timing has assumed that accurate measurements and interactions can only occur when the objects involved are temporally co-located, with the implication that occasional resynchronization is always required. It is also possible for measurements and interactions to be calculated from temporally disjoint objects. Of course, such calculations tend to be more complex, but with such a capability fewer resynchronizations are needed solely to make measurements or predictions about the future. However, at least partial resynchronizations will still be required when predicted interactions require other objects to jointly manifest a new behavior at a certain time (e.g. a predicted collision avoidance alert requiring two aircraft to synchronize and communicate at the start of the alert). Likewise, in a simulation with stochastic elements, such predictions can not be made with certainty and hence remain susceptible to inaccuracies.

Likewise, it has been assumed that the simulation always runs 'forwards' in time. With this assumption, conservative timing intervals are generated to avoid 'missing' any important interactions. For some applications, simulations capable of running 'backwards' to a potential missed interaction are possible, with the benefit of relaxing timing intervals[20,21]. However, these 'rollback' or 'timewarp' simulations can fit better some domains than others, as some types of models are easier to either 'run' backwards or store their recent state-space so that the simulator can be backed up to before the missed problem (such methods have most commonly been applied to systems with purely discrete dynamics or very simple continuous time models). Likewise, these methods incur a computational hit, and so should be used wisely.

CONCLUSION

This paper has discussed issues with simulating large, complex systems as an analysis method during their design. Hybrid-system simulation is an emerging field of interest with the potential to provide such an analysis tool.

Simulation of the NAS for safety analysis was used as an example and a test-case throughout the paper. This application shares many of the qualities (and requirements) of other aerospace systems. For example, large-scale simulations of many operational

systems are now being proposed, including military mission planning and spacecraft Launch and Range Operations (LRO). Likewise, detailed of a single vehicle's avionics systems now presents a complex analysis task of both aircraft dynamics and discrete transitions in mechanical, electronic and software on-board systems.

However, several open issues remain with hybrid-system simulation. Some can be addressed by a software architecture specifically designed to allow this purpose. This paper suggested that such an architecture should place few restrictions on the types of models allowed, so that it can be used for a variety of purposes and with components of varying fidelity and resolution.

The behavior and performance metrics of hybrid-systems both rely on interactions between individual components. As such, a simulation architecture also needs to accurately capture and/or create these interactions.

Methods of making the simulation as computationally efficient as possible are important. Rather than reducing the need for computational efficiency, recent improvements in computational power have, for the first time, allowed the research community to hope that very large, very complex systems can be simulated. As these simulations become more widely used, there may be increasing demand for more fidelity, more accuracy, and for more simulation runs in an analysis for wider or more statistically verifiable results.

This paper concentrated of the use of timing the updates of individual objects within a large-scale simulation. Two specific mechanisms were discussed: variable time-steps, and asynchronous simulation with occasional synchronization to capture measurements and interactions.

As a test case, a simulation architecture was discussed which met the requirements and mechanisms discussed in the paper. This simulation architecture uses an object-oriented framework to accept objects of a wide variety of types, easily incorporating both continuous time and discrete event models. This simulation architecture was used to simulate the dynamics of the National Airspace System (NAS). Even the simple methods of improved simulation timing were found to have significant benefits in this application.

ACKNOWLEDGEMENTS

This work was funded by the NASA Ames Research Center under Grant NAG 2-1291, with Irv Statler, Mary Conners and Kevin Corker administering and serving as technical points of contact. The authors would also like to thank the people who have contributed to the development of the simulation, including Serhan Ziya, Corey Ippolito and Ted Chen.

REFERENCES

[1] Saleh,R., Jou, S.J., and Newton, A.R. (1994) *Mixed-Mode Simulation and Analog Multilevel Simulation*, Kluwer, Boston.

[2] Odoni, A.R. et al (1997) "Existing and Required Modeling Capabilities for Evaluating ATM Systems and Concepts", Technical Report, MIT International Center for Air Transportation.

[3] Jim, H.K., and Chang, Z.Y. (1998) "An Airport Passenger Terminal Simulator: A Planning and Design Tool", *Simulation Practice and Theory*, p. 387-396.

[4] Kheir, N.A. (1996) "Continuous-Time and Discrete-Time Systems", *Systems Modeling and Computer Simulation*, Ed. N.A. Kheir, Marcel Dekker, New York NY.

[5] Beltrami, E. (1987) *Mathematics for Dynamic Modeling*, Academic Press, Boston.

[6] Press, W.H., Flannery, B.P., Teukolsky, S., and Vetterling, W.T. (1992) *Numerical Recipes in C – The Art of Scientific Computing*, 2nd Ed., Cambridge University Press, New York NY.

[7] Fahrland, D.A. (1970) "Combined Discrete Event Continuous Systems Simulation", *Simulation*, p. 61-72.

[8] National Research Council (1997) *Flight to the Future: Human Factors in Air Traffic Control* Ed. C.D. Wickens, A.S. Mavor, and J.P. McGee. National Academic Press, Washington DC.

[9] Hanke, C.R. (1971) "The Simulation of a Large Jet Transport", NASA Contractor Report CR-1756, Volumes 1 and 2.

[10] Johnson, E.N. and Hansman, R.J. (1994) "Multi-Agent Flight Simulation with Robust Situation Generation", MIT Aeronautical Systems Laboratory Report ASL-95-2, MIT, Cambridge MA.

[11] Stevens, B., and Lewis, F. (1992) *Aircraft Control and Simulation*, John Wiley, New York NY.

12 Friedman, L.W. (1996) The Simulation Meta-Model, Kluwer, Norwell MA.

[13] Wieting, R. (1996) "Hybrid High-Level Nets", *Proceedings of the 1996 Winter Simulation Conference*, p. 848-855.

[14] Cellier, F.E. (1986) "Combined Continuous/ Discrete Simulation Applications, Techniques, and Tools", *Proceedings of the 1986 Winter Simulation Conference*.

[15] Kettenis, D.L. (1997) "An Algorithm for Parallel Combined Continuous and Discrete-Event Simulation", *Simulation Practice and Theory*, Vol. 5, p. 167-184.

[16] Bezdek, W.J., Halley, T.A., and Hummel, P.C. (1997) "Model Reuse for Software Development and Testing: The Application of Common Interfaces to Support Variable Fidelity Models", *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, New Orleans LA, p. 376 – 386.

[17] Davis, P.K. and Bigelow, J.H. (1998) "Experiments in Multiresolution Modeling (MRM)", Report MR-1004-DARPA, Rand, Santa Monica CA.

[18] Pritchett, A.R. and Ippolito, C. (2000) "Software Architecture for a Reconfigurable Flight Simulator" *AIAA Modeling and Simulation Technologies Conference* Denver CO.

[19] Law, A.M. and Kelton, W.D. (1991) *Simulation Modeling and Analysis*, 2nd Ed, McGraw-Hill, New York NY.

[20] Mirtich, B. (2000) "Timewarp Rigid Body Simulation" *SIGGRAPH 00.*

[21] Jefferson, D.R. (1985) "Virtual Time" *ACM Transactions on Programming Languages and Systems*, Vol. 7, No. 3, p. 404-425.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37244

AIAA-2000-4186

GENERIC DEVELOPMENT ENVIRONMENT FOR INS ALGORITHMS

Jacob Reiner
RAFAEL, Department of Development and Engineering
P.O. Box 2250, Haifa 31021, Israel.

Abstract

This paper discusses the Generic Development Environment for INS Algorithms (GDEIA) currently been developed in the RAFAEL Navigation Group. The development of the GDEIA was motivated by system requirements, including low-cost, reusability, reliability, commonality and portability, that should be achieved during the algorithm development process and subsequent verification steps. These requirements can arguably only be met with the help of a user-friendly and powerful development framework. The GDEIA appears to be an adequate candidate for that purpose.

1. Introduction

A modern Inertial Navigation Systems (INS) is a combination of hardware for taking measurements and software for transforming these measurements into a navigation solution. The basic hardware of an INS is called an "Inertial Measurement Unit" (IMU), and includes a triad of accelerometers for measuring linear accelerations and of gyroscopes for measuring angular velocities. The basic software of almost every modern INS is a strapdown algorithm; this algorithm integrates the output signals from the gyroscopes and the accelerometers, to compute the attitude, position and velocity solutions.

The fact that the IMU outputs are integrated to produce a navigation solution implies that the quality of the latter is almost completely dictated by the quality of the IMU. As a consequence, applications requiring an accurate navigation solution would in principle require relatively high performance IMUs. Since the cost of an IMU escalates rapidly as performance increases, accurate solutions would in principle require an expensive INS. Here "expensive" includes not only the price itself, but also other parameters like volume, weight and power consumption. To circumvent this difficulty, it is possible to introduce external measurements or "aiding," in the attempt to increase performance without a substantial increase in price. The Global Positioning System (GPS), and Doppler or barometric altitude measurement instruments are typical examples of aiding to the basic INS solution.

The inclusion of aiding allows the use of less expensive IMUs at the price of increasing the overall system complexity. This increase in complexity is both in terms of hardware (the aiding instrument itself plus the necessary overhead to make the measurements available to the navigation computer) and software (for fusing the new measurement into the navigation solution.) In an aided INS, navigation performance is a function of the IMU quality and of the algorithms/data processing implemented by the navigation software.

Among the many properties a practical INS should have, it should meet at least the following:

- Performance requirements.
- Robustness to variations in the nominal design.
- Cost effectiveness.
- Low-cost maintenance.
- Portability between different platforms.

The development process of the INS must be aimed at satisfying the above requirements subject to the goals of the project in which the INS is part of. It is clear that an increase in system complexity affects the development process in several manners.

One of the main issues that should be addressed during the development process of a complex system is the one of testability. The development process should, at each development step, put an emphasis on the verification test of each item. The purpose of these a-priori considerations is to make the verification phase as short as possible, while at the same time keeping the cost low and the results reliable. The requirements for shortening development time, reducing cost and keeping the testability to be reliable bring up the necessity of a generic development process, capable of conciliating these three contradictory aspects.

The main activity of the RAFAEL Navigation Group is the integration of hardware acquired from different manufacturers with RAFAEL homemade software, essentially based on the RAFAEL homemade navigation algorithms. In the face of the previous paragraph, substantial effort has been spent onto

Copyright © 2000 by RAFAEL. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission

providing a design and development environment capable of achieving the above objectives. This paper discusses the results of these efforts, presenting what we have called the "Generic Development Environment of INS Algorithms" (GDEIA). The aim of the GDEIA is to facilitate the software development stage and to allow a smooth transfer of the software to all projects.

As mentioned above the GDEIA development was motivated from the development process and the tests of the navigation algorithms. Hence, paragraph 2 introduces the navigation algorithms development process and paragraph 3 introduces the algorithms verification tests. Paragraph 4 presents the GDEIA and the conclusions are summarized in paragraph 5.

2. Navigation Algorithm Development Process

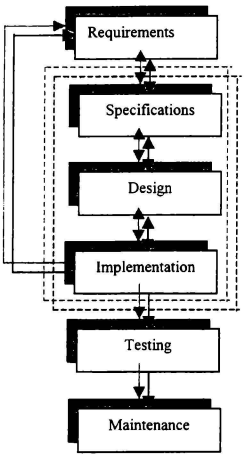


Figure 1: Waterfall Model of the Development Process

The INS development process can be modeled as the cascade process illustrated in Fig. 1, known as the waterfall process. As shown in the figure, in order to obtain an operational INS, the navigation algorithms developed at the Design stage by the Navigation Engineer should be coded by a Software Engineer and tested on the Real-Time (RT) target processor. In a traditional development process, the software engineer waterfall process could be enabled only after the navigation engineer went through his waterfall process up to his algorithms verification tests. Even more, the navigation engineers were using one high-level programming language (usually FORTRAN) and the software engineers were using usually a different programming language (like Assembler, PLM, Pascal, ADA or C). Fig. 2 shows a schematic representation of the traditional way of developing navigation algorithms and the corresponding software. It is clear from the diagram that there is almost no overlap between the navigation and the software engineers. The lack of active interaction between the two disciplines (navigation and software) created a major waste of time, resources and money. At every verification stage, at the system level, both engineers have to be presented because there could not be any responsibilities or capabilities transferred in-between both engineers.

In order to reduce these extra expenses, a new design paradigm was developed at the Rafael Navigation Group. This paradigm is based on the premise that a larger overlap between the algorithm design and the algorithm implementation stages, if appropriately organized and controlled, would result on a more efficient development cycle. The new paradigm is illustrated in Fig. 3. This paradigm, contrary to the previous one, enables capabilities and responsibilities sharing between the navigation engineer and the software engineer. Hence, only one engineer should be presented at the system level verifications.

The GDEIA presented in this paper makes the interaction between the navigation people and the software people much easier and faster. The common inputs/outputs to/from all the navigation algorithms in all projects of all directorates makes things more clear for both.

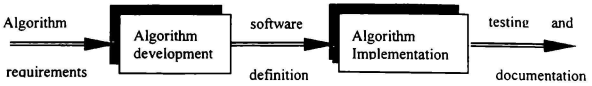


Figure 2: Model of traditional development paradigm

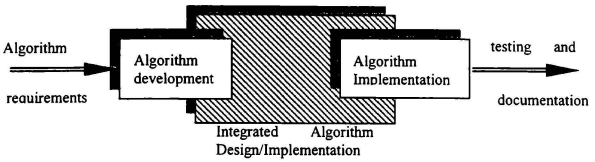


Figure 3: Model of new development paradigm

3. Verification Tests of the Navigation Algorithms

A navigation algorithm must pass the following four verification tests before it can be considered operational:

1. Algorithm verification.
2. Software verification on the target processor
3. Integration verification.
4. Overall system verification.

Figure 4 illustrates a typical environment for the development and test of navigation algorithms. The Trajectory Generator block generates two different files: a sensor data file and an aiding data file. The sensor file contains the measurements that would have been generated by the IMU along the trajectory generated by the Trajectory Generator. The aiding data file contains the measurements that would have been generated by the aiding devices (e.g., GPS, Aircraft INS, barometer, Doppler, etc.) running the navigation simulation. This distinction between "sensor" and "aiding" files is rather artificial and reflects the fact that the inertial solution is considered to be the basic ones, while the remaining measurements are used to decrease the errors as much as possible. More generally, the Trajectory Generator will produce one or more files containing the simulation of the output of any relevant instrument.

It is worth stressing that these files contain ideal sensor readings. In order to test the performance of the navigation algorithm, the measurements should be contaminated by measurement error modeled after the noise expected from the instruments. As shown in Figure 4, measurement errors are added to the measurements in the Navigation Simulator. This allows the user to test the performance of the algorithm over the same trajectory but under different measurement error conditions. This is important for two reasons. First, the sensitivity of the algorithm with respect to specific parameter variations can be investigated. Second, the parameters of the error models can be selected at random and hence Montecarlo runs can be performed in a straightforward and computationally efficient manner.

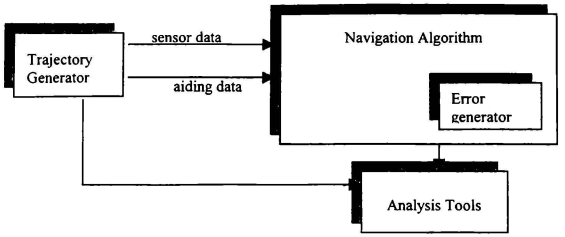


Figure 4: Setup for Algorithm Verification

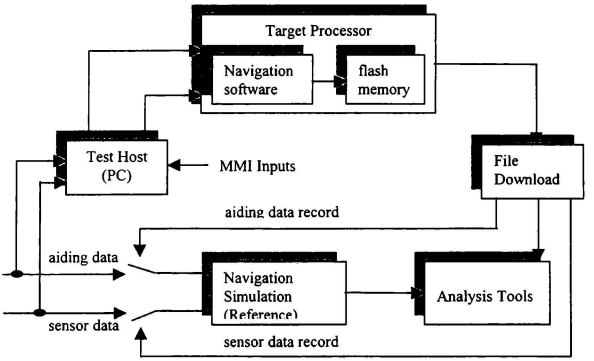


Figure 5: Setup for Software Verification

The output of the navigation algorithm, both for individual runs and for statistical results obtained from a Montecarlo simulation, is tested against the expected performance, computed using simplified analysis models and a set of specifications.

The environment for testing the navigation Real-Time (RT) software on the target processor is shown in Figure 5. It is important to notice that the RT navigation algorithm should be identical to the navigation algorithm used for simulation; the latter can then be used as a reference for the former. This identity should be carried as far and deep as possible, including not only functions and timing, but also even names for the variables and functions.

The test equipment imports the files containing simulated sensor and aiding measurements, which are created by the Trajectory Generator block as described above. A special application runs in the test equipment processing the following data:

1. Sensor data from the files created by the Trajectory Generator.
2. Reference data (to be used as aiding data) from the files created by the Trajectory Generator.
3. Status and general simulation setup, generated by a Man Machine Interface (MMI).
4. Error data for the sensors and the aiding devices generated by the MMI.
5. Timing and scheduling of events also generated by the MMI.

The application transfers the processed data to the target hardware through the communication port (e.g. 1553, RS-422 or RS-232) using the protocol defined for that purpose in an Interface Control Document (ICD). While running in Real-time mode, the manager task of the RT software writes the data required for off-line analysis in flash memory. After the conclusion of the RT run, the files required for off-line analysis are downloaded from the flash memory to the hard disk of the computer used for the analysis of the results. Two different off-line procedures are performed on the RT data to evaluate the performance of the RT software, namely **Data Analysis** and **Data Reconstruction**. In the Data Analysis step, the records from the real-time run are processed, presented graphically and studied using special purpose software. The analysis of the RT results can be done both with respect to the results obtained at the simulation stage or as stand-alone, using the same data files used for the test. In the Data Reconstruction step, the trajectory used during the RT computations is fed into the navigation simulation. Subsequently, the RT results are compared with the ones obtained from the off-line simulation. Up to unavoidable real-time nuisances, the two sets of results should be identical.

The integration verification test is performed similarly to the software verification stage, by replacing simulated sensors by real hardware, subject to the availability of the necessary facilities. For instance, a simulated IMU can be replaced by a real IMU, and a simulated GPS receiver replaced by a real GPS receiver. Notice that using real equipment may limit the scenarios and operation points that can be tested during this stage. It is of course good engineering practice to integrate the real hardware gradually into the system, usually one component at a time, until all components are integrated, tested and certified. Data downloading and analysis should be performed in the same way as in the software verification test step described above.

After passing the integration verification test, the integrated system is ready for use under operational conditions, which should eventually result in the overall system verification. Once more, the software tools developed for the previous tools can be used for evaluating the system performance. Commonality and compatibility should be sought at each and every stage to maximize software re-utilization.

In correspondence to current usage, the files containing the simulation of physical and dynamic data, that serve as inputs to the different simulators (e.g., development simulation, test equipment simulations, hybrid simulation, training simulation), should be generated by the navigation group. The data is exported to the test equipment as text/binary files or as dynamic link libraries (DLL's). The former case corresponds to navigation-only, open loop tests, while the latter is for system closed-loop tests together with corresponding guidance and control modules. Since the navigation development tools affect the testability of the overall system, the navigation development process should, from its very beginning, take into account the future tests at all levels of integration of the project.

The work of the navigation group in RAFAEL is oriented by the ideas discussed in the previous paragraphs, and emphasis is placed on designing all the navigation systems as variations from a single source. This philosophy enables the drive towards generic RT navigation algorithm software, together with generic software for equipment testing and hybrid simulations. The software derived from the generic RT navigation and sensor simulation has the following advantages:

1. Easy to reuse. The core navigation software is used by different projects. The navigation engineer can then concentrate on the specifics of his/her project.
2. Portable. The fact that the software is generic implies that additional efforts have been made to make it portable to different hardware configurations and operating systems.
3. More reliable. The fact that the software has been used and verify before implies that at least the core algorithm is reliable and (hopefully) bug-free.

4. Easy to manage. All software is stored in a database, thus facilitating software management and version control.

The development of a navigation algorithm under the GDEIA is outlined in the next section.

4. The Generic Development Environment of INS Algorithms

The purpose of this section is to describe the GDEIA from the point of view of a user (i.e., a navigation engineer). As mentioned in the introduction, the traditional development process illustrated in Figure 2 was quite tedious and resource consuming. As opposed to this, the code developed for simulations under the GDEIA is identical to the one ultimately implemented on the target processor. All the coding, documentation and verification, up to the unit test level and at the performance level, are lead by the navigation engineer in full cooperation with the RT software engineer.

In the real-time application, the navigation algorithm or "NavSeg" interfaces with the navigation instruments and aiding-devices through software drivers. In simulation, the drivers are replace by an application called Real-Time Frame (RT-Frame); the RT-Frame accepts the data from a file as input and generates the data to be imported into the NavSeg as output. Note that the navigation group is responsible for the development of the mathematical algorithm in the NavSeg, from definition and up to final verification, as well as for the verification of all the RT handlers that create the inputs needed by the navigation algorithm. The architecture of the GDEIA is shown in Figure 6.

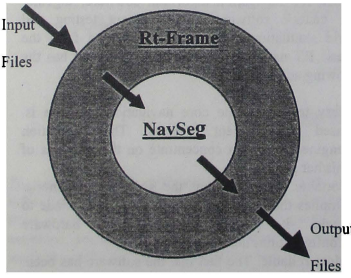


Figure 6: General Development Environment of INS Algorithms (GDEIA)

The RT handlers are a set of software objects simulating the interface of the NavSeg with the hardware. This includes functions for analyzing the output data from the simulated hardware and for transforming this data to physical units (as expected by the NavSeg). It is worth stressing that although the interface changes, the NavSeg is the same for both simulations and implementations. The input files to the simulation are imported from either one of the following three possible sources:

1. The output of a trajectory generator and sensor/aiding simulation.
2. Real-time data, extracted from recorded file.
3. The output of a Six Degrees-of-Freedom simulation.

Although the sources vary, the data format should be identical in all cases. Indeed, the objective of having a generic RT-Frame common to all projects implies the need of standardizing the input and output files. Appendix A shows an example of this standardization: no matter which GPS receiver is finally used in an application, nor which algorithm is implemented in the NavSeg, the data-structure imported into the NavSeg is the same. This is an important difference with previous usage, where more often the data structure were defined depending on the specific necessities of a project.

A similar remark can be made concerning the output files. For example, a pre-defined and standard output structure is shared by all NavSeg algorithms including a GPS/INS Kalman Filter. This standardization in all input/output files and structures make the analysis tools, Interface Control and documentation easy to transfer from project to project.

In the Rafael Navigation Group, the RT-Frame was implemented using object oriented technology, with C++ as the programming language. The RT-Frame is composed of several basic building objects. Typical simulated measurements include: Inertial Measurement Unit, aircraft INS, GPS receiver, barometer, and Doppler. Every object is self-contained, in the sense that it includes all properties and actions associated with the particular sensor. For example, the following properties are built into the IMU object: It has 3 gyroscopes, 3 accelerometers, an error model and the corresponding error parameters. The following actions are defined for the IMU object: read IMU measurements from a file with a specific data structure, computes measurement errors, add the errors to the raw

measurements and export the data to other objects with a specific structure,. The same methodology is implemented with other objects as well.

The interaction of the navigation engineer with the GDEIA is through a Man Machine Interface (MMI) programmed for that purpose using the Microsoft Developer Studio©. The MMI is a menu-driven application, which allows the user to tailor his or her simulation and test to the needs of the project. The main menu is shown in Figure 7. The MMI has three parts: one dedicated to the environment, the second to the Error Sources and the third to General specifications.

In the **Set Environment Section** the user can specify the directories for input/output files, together with their respective names. The menu also allows the selection of several frequencies relevant to the simulation, like strapdown calculation frequency, aircraft update frequency and GPS update frequency. This menu also allows the selection of measurement delay as an integer multiple of the strapdown frequency. The default input data is the data of last simulation. Finally, the menu allows the specification of a title for the specific simulation and keeps the simulation input/output data in a file with name derived from the simulation general name.

One of the remarkable features of the MMI is that it includes a direct access to the parameters of the model for the navigation instrument errors described above. This models can be used together with the nominal data of the sensors and aidings, read from the input files, for generating realistic navigation instrument outputs, and hence, examine the behavior of the algorithm under development working on the resulting data. The MMI allows both for fixed parameters or for randomized data for Montecarlo simulations. The following are some of the error sources used in the GDEIA:

IMU Errors:

1. **Accelerometers errors**: long term bias, bias short term instability (first order Markov process), scale factor, scale factor asymmetry, scale factor non-linearity, misalignment, vibration rectification, random walk.
2. **Gyroscopes errors**: long term drift, drift short term instability (first order Markov process), scale factor, scale factor asymmetry, scale factor non-linearity, misalignment, aniso-elasticity, random walk.
3. **Environmental errors**: captive flight or free flight G RMS.

Aircraft INS Errors:

1. **Navigation errors:** position, velocity and attitude errors.
2. **Measurement errors**: velocity random measurement errors, attitude random measurement error, time-tag error (constant + random), aircraft lever arm error.

GPS Errors:

1. **Navigation errors**: position and velocity errors.
2. **Pseudo-range errors**: selective availability, clock bias, clock bias drift

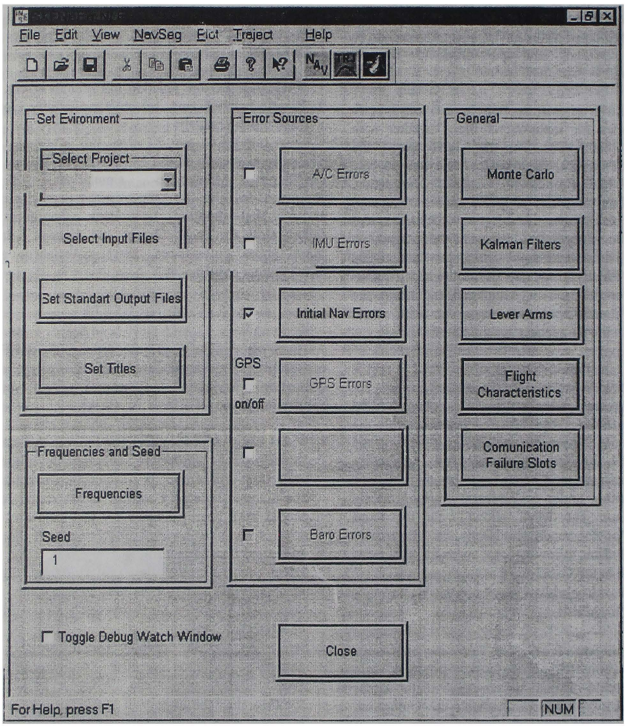
The general section of the MMI contains menus for specifying Montecarlo simulations, flight characteristics (align start time, captive flight duration and free flight duration), lever arms, communication failure slots for each element and Kalman Filters initial states covariances, noise covariances and measurement covariances.

5. Using The GDEIA

As explained in the previous sections, the GDEIA constitutes a powerful tool for designing and evaluating navigation algorithms. A typical application of the GDEIA is as follows.

1. Select a nominal trajectory. The input to this stage is a set of parameters defining the desired trajectory. The output is a set of files containing the nominal trajectory and the measurements that would have been produced for that trajectory by ideal sensors and aiding instruments. A file is generated for each sensor.
2. Select names for the project and flight characteristics. The input and output files are selected through the MMI together with the flight characteristics.
3. Select a specific project. Each project is associated with a specific configuration of sensing/aiding instruments. When a project is selected, the options related to the project can be accessed and changed one-by-one. For instance, gyroscope drifts can be selected, or barometer bias can be changed.
4. Select initial conditions. This initial conditions refer to parameters included in the filters implemented in the simulation.

5. Run a simulation. Once the parameters in 1-4 above have been selected, the "NAV" button in the MMI can be pressed to start a navigation simulation. The software associated with the MMI then processes the nominal data constructed in step 1 under the conditions selected in 2; these data is the corrupted using the errors selected in 3. The resulting data is imported to the navigation algorithm. A special window shows the user the status of the simulation.



6. Conclusions

Navigation algorithms and software development processes are in general intrinsically more complex than manufacturing processes, due to more variations, greater complexity and flexibility. Commonality of implementation and testing should come out from the expert group and not from the project. In a company, working in a matrix form, this really drives the system into deep and true commonality in the development of algorithms and test equipment.

The GDEIA described above is a tool trying to bring the commonality up to the projects. The GDEIA automates the algorithm development process so that:

- Process consistency is improved
- Time and cost of the algorithm development process is dramatically reduced
- A lot of the errors elimination activities do not exist
- Algorithms' reliability is much higher
- Algorithms' reusability is easy. Reusability reduces cost, schedule and risk
- The interface to the user is very friendly
- Configuration control of the algorithms is easier
- The same input data is being used for the development phase as well as for all the verification steps
- Modifications in the development environment of one project is easily transferred to all projects
- Adding new objects is easy.
- Creating new projects, consisting of different objects, is very easy.
- Analysis tools, interfaces and documentation commonality are being kept across the projects.

ACKNOWLEDGEMENTS

I would like to thank Mrs. Miriam Shaked for the effort she puts to build up the GDEIA, to maintain it and to force the engineers working under the generic environment. I would like also to thank Dr. Hector Rotstein for helping me writing the paper, bringing up ideas for the paper and helping us debugging the new working environment.

APPENDIX A:
Example of GPS Data-Structure

The following lines taken from a prototype GPS driver illustrate the structure defined for GPS measurement input.

```
struct GPS_PV_Meas_Struc {
    double  GPS_position [3];
    double  GPS_velocity_enu [3];
    double  clock_bias;
    double  clock_bias_rate;
    double  GPS_pos_system_time;
    double  GPS_vel_system_time;
    boolean GPS_pv_valid;
};
struct GPS_PR_Meas_Struc {
    double  SV_position     [NUMBER_OF
            _GPS_ CHANNELS][3];
    double  SV_PR     [NUMBER_OF_GPS_
            CHANNELS];
    double  Var_PR    [NUMBER_OF_GPS_
            CHANNELS];
    int     SV_id     [NUMBER_OF_GPS_
            CHANNELS];
    double  GPS_PR_system_time
            [NUMBER_OF_GPS_
            CHANNELS];
    boolean GPS_PR_valid;
};
struct GPS_Figure_Of_Merits_Struc{
    double  GPS_PDOP;
    double  GPS_HDOP;
    double  GPS_VDOP;
    double  GPS_TDOP;
    double  GPS_GDOP;
    boolean GPS_DOP_valid;
    double  GPS_FOM;
    double  GPS_TFOM
    boolean GPS_FOM_valid;
};
struct  Mgr_Nav_GPS_Data_Struc {
    unsigned long     measurement_counter;
    struct GPS_PV_Meas_Struc
                      GPS_pv_meas;
    struct GPS_PR_Meas_Struc
                      GPS_PR_meas;
    struct GPS_Figure_Of_Merits_Struc
                      GPS_fom;
};
```

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37245

AIAA 2000-4188

A SIMULATION MODEL OF AN UNINHABITED AIRCRAFT

Mario G. Perhinschi*
Flight Dynamics and Control Laboratory
National Aerospace Research Institute
Bucharest, Romania

ABSTRACT

The purpose of this paper is to present the general conceptual frame of a simulation package for the flight of an uninhabited aircraft that is currently under development as part of a virtual design environment at the National Aerospace Research Institute of Bucharest, Romania. The major components are defined according to the tasks that need to be performed. Their structural architecture, content, interconnecting relationships and informational flow are identified and described. General dynamic equations of motion of a rigid body airvehicle with rotating parts and variable center of mass position are derived using Kane's method. A partial, preliminary implementation has been performed and numerical results are presented to illustrate it.

1. INTRODUCTION

Uninhabited Aerial Vehicles (UAV) have the capability to allow humans indirect access and activity in remote, inaccessible and/or dangerous locations. The need for sensing and commanding position in minimally controlled and inaccessible environments, together with the inclusion of artificial intelligence elements and decision making capabilities raise specific problems of modeling, simulation, algorithm development and system design. Design and development of such complex systems require computer simulation as indispensable tools along the process.

This paper describes the general conceptual frame of a simulation package for the flight of an uninhabited aircraft that is currently under development, as part of a virtual design environment, at the National Aerospace Research Institute of Bucharest, Romania. Various aspects regarding structural and functional architecture, task complexity and flexibility, modeling issues and options are presented to illustrate the current status of the project along the lines of an object oriented methodology[1]. Although developed with UAVs in mind, the simulation package can accommodate conventional manned aircraft as well.

The general simulation package consists of six major modules: an Input Module, an Output Module, the General Vehicle Module, the Auxiliary Systems Module, the Mission Equipment Module and the Decision Module. Each module is described with emphasis on main structural and functional characteristics, on the interconnections with other components and on the general modeling approach. The general dynamic equations of motion of the airvehicle as a rigid body are derived using Kane's method[2]. Effects of rotating parts and of variable center of mass position are considered.

2. GENERAL SIMULATION PACKAGE

The General Simulation Package is composed of a Simulation Nucleus, an Input Module and an Output Module (see Figure 1). The Simulation Nucleus includes four major modules: the General Vehicle Module, the Auxiliary Systems Module, the Mission Equipment Module and the Decision Module.

The Human Manager will provide a priori all the necessary data to fully characterize the aircraft and other systems together with any custom routines that may be required. For a given aircraft, he/she will provide the complete menu of compatible simulation options according to objective requirements and availability of data and/or routines.

The Operator, or user, interactively defines the simulation process by providing information regarding: the aircraft identification (name or code), general simulation type, simulation sophistication level, specific objective or mission, general mission conditions, mission details.

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.
*Senior researcher, Ph.D., AIAA Member

The general type of the simulation can be: autonomous flight type, ground simulator type, programmed controls, human pilot controlled type, auxiliary and connected actions. In an autonomous flight type mission the operator provides only spatial destination and general mission characteristics and objective and the intelligent vehicle elaborates and executes the entire mission program. The ground simulator type assumes on-line, real time operation of the simulated aircraft by a human pilot in a manner similar to that performed in a ground simulator. This mode can be used for human pilot behaviour studies and modeling, handling qualities

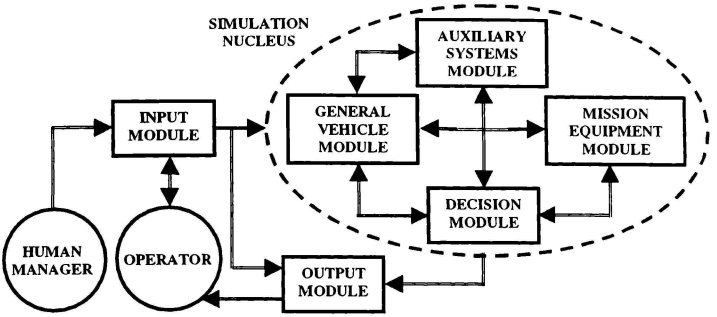


Figure 1. The Block Diagram of the General Simulation Package

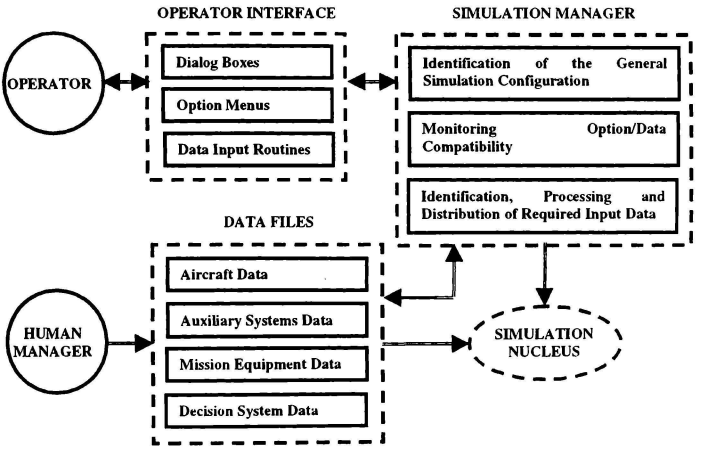


Figure 2. The Block Diagram of the Input Module

investigations, aircraft model validation. The programmed controls type simulation uses pre-loaded aircraft controls to evaluate vehicle response characteristics or investigate flight incidents and scenarios. In human pilot controlled type missions a mathematical pilot model is used to generate controls for handling qualities assessment. Auxiliary and connected actions include but are not limited to: linearization, transfer functions extraction, parameter variation, subsystem analysis, etc.

The simulation sophistication level options establish if simplified models of various subsystems can be used, if certain modules or sub-modules are to be ex/included (e.g. decision module, mission equipment module or parts of them), the general output options, the operator's authority level to intervene during the simulation.

The specific objective or mission is chosen by the operator from a menu provided by the human manager. For example, an autonomous flight type mission could be described as: "flight to a designated area, target search and identification, video recording, return to base", or "flight to a designated area, landing, payload delivery, take-off and return to base". Examples of ground simulator type missions are: take-off or landing on a ground runway, take-off or landing on a carrier deck, target acquisition and weapon delivery. Programmed controls type missions include: response to standard control inputs (steps, doublets, sinusoids, frequency sweeps), response to a specific

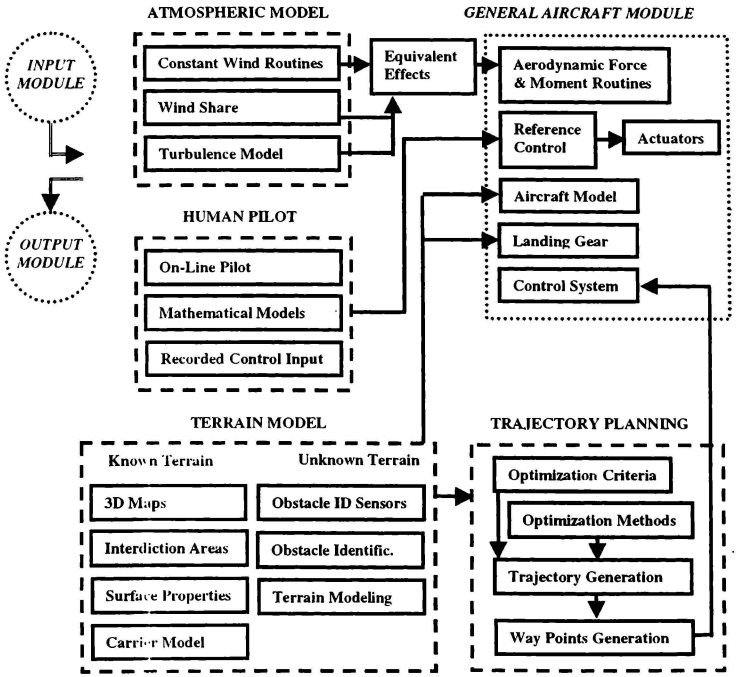


Figure 3. The Block Diagram of the Auxiliary Systems Module

control program (for performance evaluation or accident investigation).

The general mission conditions are characteristics that determine environment and other external interventions. They include, but are not limited to: presence or absence of constant wind and turbulence, complete knowledge of terrain or not, failure of systems occurring during the mission, enemy actions, various restrictions.

The mission details quantitatively define options previously mentioned. Information like wind velocity and turbulence intensity, simulation total and partial duration, destination location, flight reference velocity go in this category. Detailed output options are also included here.

3. INPUT MODULE

The Input Module includes the *Operator Interface*, the *General Aircraft and Systems Data Files*, and the *Simulation Manager* (see Figure 2).

The operator interface consists of all the software elements that allow user-program interaction for simulation definition and initialization.

The general aircraft and systems data files contain all the pre-loaded information that is necessary to correctly perform the entire set of eligible simulation configurations for a particular aircraft. They are provided by the simulation human manager.

The simulation manager sub-module prepares the basic simulation frame according to operator's simulation options. It identifies the simulation configuration, monitors the compatibility between operator's interactive options and available data and then processes the data and distributes them to the modules of the simulation nucleus.

4. AUXILIARY SYSTEMS MODULE

The Auxiliary Systems Module includes the *Atmospheric Model*, the *Trajectory Planning Sub-Module*, the *Terrain Model* and the *Human Pilot Sub-Module* (see Figure 3).

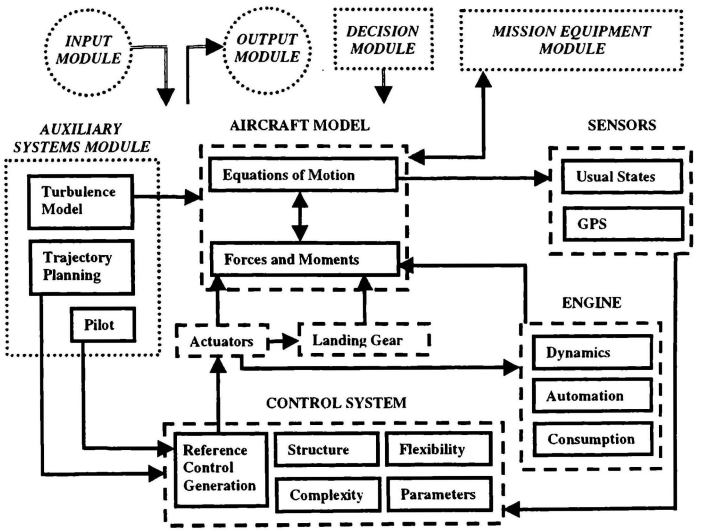


Figure 4. The Block Diagram of the General Vehicle Module

The atmospheric model addresses the flight in constant wind, flight in wind share and flight in the presence of random turbulence. The turbulence field is assumed to be homogeneous, stationary, isotropic and frozen[3]. Turbulence velocity at any point is the superposition of a random value constant over the entire vehicle and a spatial gradient. Effects of the constant turbulence field are considered as random perturbations of the air velocity. Spatial gradients effects on all parts of the aircraft but the rotor of the rotorcraft are modeled as equivalent rotations. Effects on the rotor are modeled as equivalent perturbations of the main rotor controls[4]. Spatial gradients effects are negligible for model aircraft but become significant for full-size vehicles. The method of the sum of sinusoids[5] is used to generate random air velocity perturbations that match the statistical properties of the von Karman model[6].

The trajectory planning sub-module is active for autonomous flight type missions. This sub-module computes the desired trajectory subject to an optimization process unless the trajectory is not imposed by the operator. It also provides waypoints (position and velocity) as input to the on-board controller.

There are two modes of operation for the terrain model sub-module. Flight in totally known terrain assumes existence of 3D maps that describe natural and artificial obstacles on which interdiction areas are marked that are due to enemy activity and/or other safety related causes. Obstacles and interdiction areas, modelled as path constraints, are used in the trajectory optimization/computation process. Runway surface properties and carrier deck movement models to be used for landing and take-off simulations are also part of the terrain model. Flight in totally or partially unknown terrain requires that obstacles are identified, localized and evaluated during the on-going mission. It means that the desired trajectory is generated a few steps ahead while the vehicle is moving (or hovering, if necessary and possible), based on dedicated sensors information. Designing such a system is an extremely challenging task. Simulating it may be less complex if its main characteristics can be well estimated: computational delay, accuracy, correct identification probability, operational range.

The human pilot sub-module plays a role in on-line piloted simulations, in simulations that use recorded control inputs or when the controls are generated by a human pilot mathematical model.

5. GENERAL VEHICLE MODULE

The General Vehicle Module includes models of the aircraft motion, sensors, actuators, landing gear, engine and control system (see Figure 4).

The flight dynamics and kinematics model consists of the well known twelve first order, ordinary, non-linear differential equations of rigid body motion[7] (subscript B) augmented by a variable number of equations that model the components in rotational movement (subscript R) and those whose center of mass position is variable (subscript M). Derivation of the equation has been performed using Kane's method[2]. The generalized coordinates are: measure numbers of the position vector of the aircraft center of mass in inertial frame, rotation angles of body axes (F_B) with respect to inertial frame (F_E), rotation angles of rotational parts with respect to body axes and measure numbers of the position vectors of the moving centers of mass in body axes. Generalized speeds are defined accordingly. Forces and moments are computed both with analytical formulae and experimental tables. With standard notation for the vector measure numbers in body axes the dynamic equations of motion have the form:

$$\begin{cases} a_1 \cdot \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + a_2 \cdot \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \sum_{i=1}^{n_R} a_{3i} \cdot \begin{bmatrix} \dot{p}_{Ri} \\ \dot{q}_{Ri} \\ \dot{r}_{Ri} \end{bmatrix} + \sum_{i=1}^{n_M} a_{4i} \cdot \begin{bmatrix} \dot{u}_{Mi} \\ \dot{v}_{Mi} \\ \dot{w}_{Mi} \end{bmatrix} - \beta_1 = 0 \\ a_5 \cdot \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + a_6 \cdot \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \sum_{i=1}^{n_R} a_{7i} \cdot \begin{bmatrix} \dot{p}_{Ri} \\ \dot{q}_{Ri} \\ \dot{r}_{Ri} \end{bmatrix} + \sum_{i=1}^{n_M} a_{8i} \cdot \begin{bmatrix} \dot{u}_{Mi} \\ \dot{v}_{Mi} \\ \dot{w}_{Mi} \end{bmatrix} - \beta_2 = 0 \\ a_{9i} \cdot \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + a_{10i} \cdot \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + a_{11i} \cdot \begin{bmatrix} \dot{p}_{Ri} \\ \dot{q}_{Ri} \\ \dot{r}_{Ri} \end{bmatrix} - \beta_3 = 0 \\ a_{12i} \cdot \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + a_{13i} \cdot \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + a_{14i} \cdot \begin{bmatrix} \dot{u}_{Mi} \\ \dot{v}_{Mi} \\ \dot{w}_{Mi} \end{bmatrix} - \beta_4 = 0 \end{cases} \tag{1}$$

where the following notations have been used:

$$\begin{aligned} a_1 &= -m_{AC} \cdot I \\ a_2 &= \sum_{i=1}^{n_R} m_{Ri} \cdot \left[\tilde{r}^{OO_{Ri}}\right]_B + \sum_{i=1}^{n_M} m_{Mi} \cdot \left[\tilde{r}^{OO_{Mi}}\right]_B \\ a_{3\,1i} &= m_{Ri} \cdot \left[\tilde{r}^{B_{Ri}O_{Ri}}\right]_B \\ a_{4i} &= -m_{Mi} \cdot 1 \end{aligned} \tag{2}$$

$$\begin{aligned} \beta_1 = m_{AC} \cdot \left[{}^E\tilde{\omega}^B\right]_B \left[{}^E\tilde{v}^O\right]_B + \sum_{i=1}^{n_R} A_{Ri}^{(1)} + \\ + \sum_{i=1}^{n_M} A_{Mi}^{(1)} - \left[\tilde{R}_{AC}\right]_B - m_{AC} \cdot g \cdot \begin{bmatrix} L_{BE_{13}} \\ L_{BE_{23}} \\ L_{BE_{33}} \end{bmatrix} \end{aligned} \tag{3}$$

$$\begin{aligned} a_5 &= -\sum_{i=1}^{n_R} m_{Ri} \cdot \left[\tilde{r}^{OO_{Ri}}\right]_B - \sum_{i=1}^{n_M} m_{Mi} \cdot \left[\tilde{r}^{OO_{Mi}}\right]_B \\ a_6 &= -\left[\tilde{I}^{AC}\right]_B + \left[m_R \cdot \tilde{r}^{OR}\right]_B + \left[m_M \cdot \tilde{r}^{OM}\right]_B \\ a_{7i} &= -\left[\tilde{I}^{R_i/O_{Ri}}\right]_B + m_{Ri} \cdot \left[\tilde{r}^{OO_{Ri}}\right]_B \cdot \left[\tilde{r}^{B_{Ri}O_{Ri}}\right]_B \\ a_{8i} &= -m_{Mi} \cdot \left[\tilde{r}^{OO_{Mi}}\right]_B \end{aligned} \tag{4}$$

$$\beta_2 = A^{(2)} + A^{(3)} + A^{(4)} + A^{(5)} + A^{(6)} - \\ - \left[\tilde{T}_{AC}\right]_B - \left\{\sum_{i=1}^{n_R} m_{Ri} \cdot \left[\tilde{r}^{OO_{Ri}}\right]_B + \right. \\ \left. + \sum_{i=1}^{n_M} m_{Mi} \cdot \left[\tilde{r}^{OO_{Mi}}\right]_B \right\} \cdot g \cdot \begin{bmatrix} L_{BE_{13}} \\ L_{BE_{23}} \\ L_{BE_{33}} \end{bmatrix} \tag{5}$$

$$a_{9i} = -m_{Ri} \cdot \left[\tilde{r}^{B_{Ri}O_{Ri}}\right]_B \\ a_{10i} = -\left[\tilde{I}^{Ri/O_{Ri}}\right]_B + m_{Ri} \cdot \left[\tilde{r}^{B_{Ri}O_{Ri}}\right]_B \cdot \\ \cdot \left(\left[\tilde{r}^{B_{Ri}O_{Ri}}\right]_B + \left[\tilde{r}^{OB_{Ri}}\right]_B\right) \\ a_{11i} = -\left[\tilde{I}^{Ri/O_{Ri}}\right]_B + m_{Ri} \cdot \left[\tilde{r}^{B_{Ri}O_{Ri}}\right]_B \cdot \left[\tilde{r}^{B_{Ri}O_{Ri}}\right]_B \tag{6}$$

$$\beta_3 = A^{(7)} + A^{(8)} - \left[\tilde{T}_{Ri}\right]_B - \left[\tilde{r}^{B_{Ri}O_{Ri}}\right]_B \cdot \left[\tilde{R}_{Ri}\right]_B - \\ - \left[\tilde{r}^{B_{Ri}O_{Ri}}\right]_B \cdot \begin{bmatrix} L_{BE_{13}} \\ L_{BE_{23}} \\ L_{BE_{33}} \end{bmatrix} \cdot m_{Mi} \cdot g \tag{7}$$

$$a_{12i} = -m_{Mi} \cdot I \\ a_{13i} = m_{Mi} \cdot \left[\tilde{r}^{OO_{Mi}}\right]_B \\ a_{14i} = -m_{Mi} \cdot I \tag{8}$$

$$\beta_4 = A^{(9)} - \left[\tilde{R}_{Mi}\right]_B - m_{Mi} \cdot g \cdot \begin{bmatrix} L_{BE_{13}} \\ L_{BE_{23}} \\ L_{BE_{33}} \end{bmatrix} \tag{9}$$

$$A_{Ri}^{(1)} = m_{Ri} \cdot \left\{\left[E\tilde{\omega}^B\right]_B \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[\bar{r}^{OO_{Ri}}\right]_B + \right. \\ \left. + \left(2 \cdot \left[E\tilde{\omega}^B\right]_B + \left[E\tilde{\omega}^{Ri}\right]_B\right) \left[B\tilde{\omega}^{Ri}\right]_B \cdot \left[\bar{r}^{B_{Ri}O_{Ri}}\right]_B \right\} \tag{10}$$

$$A_{Mi}^{(1)} = m_{Mi} \cdot \left\{\left[E\tilde{\omega}^B\right]_B \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[\bar{r}^{OO_{Mi}}\right]_B + \right. \\ \left. + 2 \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[\bar{v}^{O_{Mi}}\right]_B \right\} \tag{11}$$

$$A^{(2)} = \left[E\tilde{\omega}^B\right]_B \cdot \left[\tilde{I}^{B/O}\right]_B^T \cdot \left[\dot{\bar{\omega}}^{Ri}\right]_B + \\ + \sum_{i=1}^{n_R} \left(\left[E\tilde{\omega}^{Ri}\right]_B \cdot \left[\tilde{I}^{Ri/O_{Ri}}\right]_B^T \cdot \left[\dot{\bar{\omega}}^{Ri}\right]_B \right) + \\ + \sum_{i=1}^{n_M} \left(\left[E\tilde{\omega}^{Mi}\right]_B \cdot \left[\tilde{I}^{Mi/O_{Mi}}\right]_B^T \cdot \left[\dot{\bar{\omega}}^{Mi}\right]_B \right) \tag{12}$$

$$A^{(3)} = \sum_{i=1}^{n_R} m_{Ri} \cdot \left[\tilde{r}^{OO_{Ri}}\right]_B \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[\bar{r}^{OO_{Ri}}\right]_B + \\ + \sum_{i=1}^{n_M} m_{Mi} \cdot \left[\tilde{r}^{OO_{Mi}}\right]_B \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[\bar{r}^{OO_{Mi}}\right]_B \tag{13}$$

$$A^{(4)} = \left\{\sum_{i=1}^{n_R} m_{Ri} \cdot \left[\tilde{r}^{OO_{Ri}}\right]_B + \sum_{i=1}^{n_M} m_{Mi} \cdot \left[\tilde{r}^{OO_{Mi}}\right]_B \right\} \cdot \\ \cdot \left[E\tilde{\omega}^B\right]_B \cdot \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{14}$$

$$A^{(5)} = \sum_{i=1}^{n_M} 2 \cdot m_{Mi} \cdot \left[\tilde{r}^{OO_{Mi}}\right]_B \cdot \left[E\tilde{\omega}^B\right]_B \cdot \begin{bmatrix} u_{Mi} \\ v_{Mi} \\ w_{Mi} \end{bmatrix} \tag{15}$$

$$A^{(6)} = \sum_{i=1}^{n_R} \left\{ \left[\tilde{I}^{Ri/O_{Ri}}\right]_B \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[\bar{\omega}^{-Ri}\right]_B + \right. \\ + m_{Ri} \cdot \left[\tilde{r}^{OO_{Ri}}\right]_B \cdot \left(2 \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[B\tilde{\omega}^{Ri}\right]_B \cdot \left[\bar{r}^{B_{Ri}O_{Ri}}\right]_B + \right. \\ \left. \left. + \left[B\tilde{\omega}^{Ri}\right]_B \cdot \left[B\tilde{\omega}^{Ri}\right]_B \cdot \left[\bar{r}^{B_{Ri}O_{Ri}}\right]_B \right) \right\} \tag{16}$$

$$A^{(7)} = \left[\tilde{I}^{Ri/O_{Ri}}\right]_B \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[\bar{\omega}^{B-Ri}\right]_B + \\ + \left[E\tilde{\omega}^{Ri}\right]_B \cdot \left[\tilde{I}^{Ri/O_{Ri}}\right]_B^T \cdot \left[\bar{\omega}^{E-Ri}\right]_B \tag{17}$$

$$A^{(8)} = m_{Ri} \cdot \left[\tilde{r}^{B_{Ri}O_{Ri}}\right]_B \cdot \left\{\left[E\tilde{\omega}^B\right]_B \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[\bar{r}^{OO_{Ri}}\right]_B + \right. \\ + 2 \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[B\tilde{\omega}^{Ri}\right]_B \cdot \left[\bar{r}^{B_{Ri}O_{Ri}}\right]_B + \\ \left. + \left[B\tilde{\omega}^{Ri}\right]_B \cdot \left[B\tilde{\omega}^{Ri}\right]_B \cdot \left[\bar{r}^{B_{Ri}O_{Ri}}\right]_B + \left[E\tilde{\omega}^B\right]_B \cdot \left[\bar{v}^{E-O}\right]_B \right\} \tag{18}$$

$$A^{(9)} = m_{Mi} \cdot \left\{\left[E\tilde{\omega}^B\right]_B \cdot \left[\bar{v}^{E-O}\right]_B + \left[E\tilde{\omega}^B\right]_B \cdot \left[E\tilde{\omega}^B\right]_B \cdot \right. \\ \left. \cdot \left[\bar{r}^{OO_{Mi}}\right]_B + 2 \cdot \left[E\tilde{\omega}^B\right]_B \cdot \left[\bar{v}^{B-O_{Mi}}\right]_B \right. \tag{19}$$

Total mass of the aircraft is:

$$m_{AC} = m_B + \sum_{i=1}^{n_R} m_{Ri} + \sum_{i=1}^{n_M} m_{Mi} \tag{20}$$

The inertia dyadic of the rigid body B with respect to the point O is $I^{B/O}$ and the corresponding inertia

matrix in frame A is: $\left[\tilde{I}^{B/O}\right]_A$. The inertia matrix of the aircraft in body axes components is

$$\left[\tilde{I}^{AC}\right]_B = \left[\tilde{I}^{B/O}\right]_B + \sum_{i=1}^{n_R}\left[\tilde{I}^{Ri/O_{Ri}}\right]_B + \sum_{i=1}^{n_M}\left[\tilde{I}^{Mi/O_{Mi}}\right]_B \quad (21)$$

The resultant force on the whole aircraft is:

$$\left[\bar{R}_{AC}\right]_B = \left[\bar{R}\right]_B + \sum_{i=1}^{n_R}\left[\bar{R}_{Ri}\right]_B + \sum_{i=1}^{n_M}\left[\bar{R}_{Mi}\right]_B \quad (22)$$

The resultant torque on the whole aircraft is

$$\left[\bar{T}_{AC}\right]_B = \left[\bar{T}\right]_B + \sum_{i=1}^{n_R}\left(\left[\bar{T}_{Ri}\right]_B + \left[\tilde{r}^{OO_{Ri}}\right]_B \cdot \left[\bar{R}_{Ri}\right]_B\right) + \sum_{i=1}^{n_M}\left(\left[\bar{T}_{Mi}\right]_B + \left[\tilde{r}^{OO_{Mi}}\right]_B \cdot \left[\bar{R}_{Mi}\right]_B\right) \quad (23)$$

We have also used the notation:

$$\left[m_R \cdot r^{OR}\right]_B = \sum_{i=1}^{n_R}\left(m_{Ri} \cdot \left[\tilde{r}^{OO_{Ri}}\right]_B \cdot \left[\bar{r}^{OO_{Ri}}\right]_B\right)$$
$$\left[m_M \cdot r^{OM}\right]_B = \sum_{i=1}^{n_M}\left(m_{Mi} \cdot \left[\tilde{r}^{OO_{Mi}}\right]_B \cdot \left[\bar{r}^{OO_{Mi}}\right]_B\right) \quad (24)$$

The position vector of point P with respect to point O has been denoted by $\bar{r}^{OP}$. The velocity and acceleration vectors of point P with respect to frame A are respectively:

$${}^{A-P}_{v} \stackrel{\Delta}{=} {}^{A}\frac{d\bar{r}^{OP}}{dt}, \qquad {}^{A-P}_{a} \stackrel{\Delta}{=} {}^{A}\frac{d\,{}^{A-P}_{v}}{dt} \quad (25)$$

The angular velocity of the rigid body B (or the equivalent reference frame B) with respect to reference frame A is ${}^{A-B}\bar{\omega}$. The angular acceleration of B in A is:

$${}^{A-B}_{\alpha} \stackrel{\Delta}{=} {}^{A}\frac{d\,{}^{A-B}\bar{\omega}}{dt} \quad (26)$$

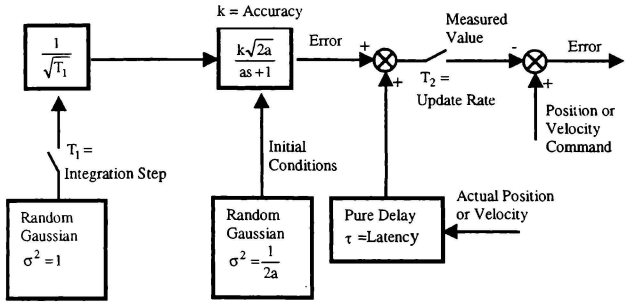


Figure 5. GPS Error Model in Position and Velocity Feedback Loops

The measure numbers of vector $\bar{v}$ in the reference frame A are denoted by:

$$\left[\bar{v}\right]_A = \left[v_1 \;\; v_2 \;\; v_3\right]^T \quad (27)$$

The measure numbers in A of the cross-product are denoted by:

$$\left[\bar{v} \times \bar{\omega}\right]_A = \left[\tilde{v}\right]_A \cdot \left[\bar{\omega}\right]_A \quad (28)$$

where the following notation has been used:

$$\left[\tilde{v}\right]_A = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \quad (29)$$

The transformation matrix from reference frame F_E to reference frame F_B is L_{BE}, hence:

$$\left[\bar{v}\right]_B = L_{BE}\left[\bar{v}\right]_E \quad (30)$$

Simple actuator models are assumed in all the four control channels as first order representations:

$$F_{act}(s) = \frac{1}{\tau s + 1} \qquad (31)$$

A simple model has been developed to generate the errors of the Global Positioning System (GPS). This model is included in the position and velocity feedback loops on all three axes. The model[8] has the same structure for both position and velocity but with different parameter values. The block diagram is presented in Figure 5. The main features of the GPS which have been considered are the latency, the update rate, the accuracy and the error dynamics parameter. The update rate represents the rate at which the position and velocity signals are sent to the receiving processor and is modeled as a quantization. The latency is the time delay that occurs between the time the satellite information is received and the time the position/velocity output is sent to the receiver. It is modelled as a pure time delay. The accuracy is the radius of the circle with the origin at the actual position/velocity which contains 50% of the sensors output values. The errors of the GPS sensors package are generated as output of a first order linear differential equation with random Gaussian input and initial conditions.

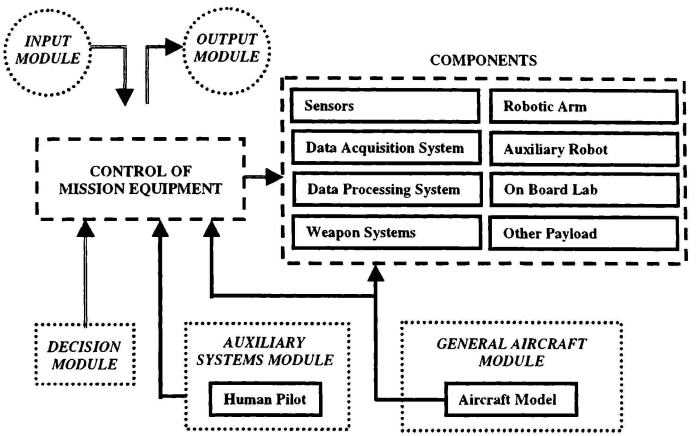


Figure 6. The Block Diagram of the Mission Equipment Module

The general architecture of the simulation model can accommodate linear, conventional nonlinear and fuzzy logic based controllers. Errors with respect to reference position and velocity components on all three axes and yaw attitude are converted into four vehicle controls (providing control moments on three axes and thrust control).

6. MISSION EQUIPMENT MODULE

The Mission Equipment Module has two parts (see Figure 6): a control sub-module and the components sub-module. The equipment that serves the purpose of the aircraft in conjunction with its mission may consist of: various sensors, data acquisition system, data processing system, weapon systems, robotic arm, auxiliary robot, on board laboratory, other payload. It should be noted that only those elements and characteristics directly related to the aircraft motion and control need to be modeled and not the specific functioning of the mission equipment systems. For example, if the aircraft is equipped with a video camera on a moving platform only the control and movement of the camera have to be modeled, not the recording process itself. The duration of the recording may also be a constraint to the motion of the vehicle in some scenarios.

7. DECISION MODULE

The decision module (see Figure 7) incorporates the most important features of artificial intelligence of an autonomous airvehicle. It hosts the following processes: aircraft and mission status evaluation, failure detection and counter measures, mission redefinition according to system failure, environment changes or operator intervention. The decision algorithms incorporate a certain level of authority and multi-criterial strategies. Artificial intelligence techniques, like fuzzy reasoning, machine learning and neural networks concur to perform this task.

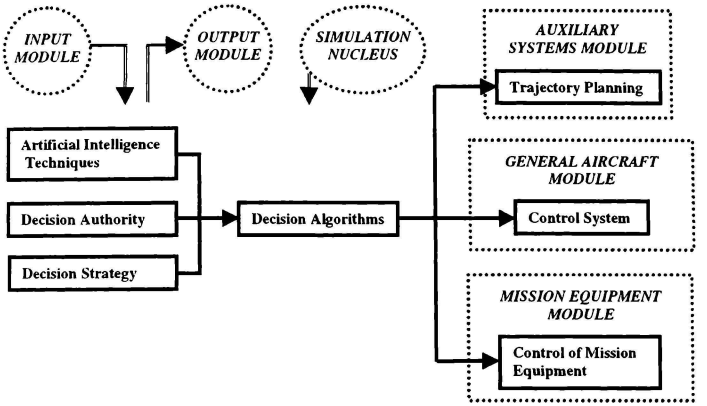


Figure 7. The Block Diagram of the Decision Module

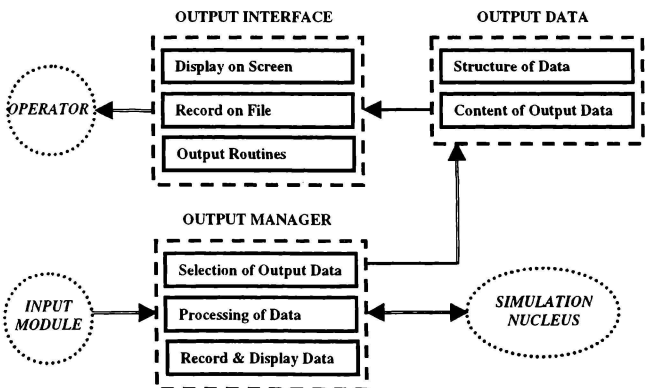


Figure 8. The Block Diagram of the Output Module

8. OUTPUT MODULE

The Output Module includes the *Output Interface*, the *Output Manager* and the *Output Data* themselves (see Figure 8).

The Output Manager organizes the selection, processing, storage and displaying of the data according to operator's options.

The content of the output data is the result of the numerical simulation and their structure can be either pre-determined or specifically requested by the operator.

The output interface consists of all the software elements that allow the program to present, display and store the numerical results of the simulation according to the users needs and options.

9. NUMERICAL RESULTS

A preliminary simplified version of the simulation package has been implemented using MATLAB. It includes the input/output modules, the general aircraft module and the auxiliary systems module.

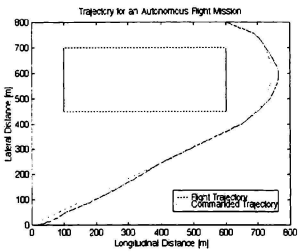


Figure 9. Commanded and Actual Trajectory in an Autonomous Flight Mission

Numerical tests have been performed for a reduced size model airplane in autonomous flight. The operator provides coordinates of the destination, the mission required is simply to reach that point with a prescribed velocity vector. The trajectory planner determines the desired flight path with a simplified algorithm based on minimum distance on a bidimensional map. A fuzzy logic based controller has been modeled with the following main characteristics: trapezoidal shape of the membership function; defuzzification method: Height Method associated with peak value; number of linguistic values: 5. Scaling factors on each of the four control channels have been determined by means of a genetic algorithm[9] that maximizes a weighted multi-criterial performance index based on time response parameters.

The commanded and actual trajectories are illustrated in Figure 9.

10. CONCLUSIONS

The general conceptual frame of a simulation package for the flight of an uninhabited aircraft has been defined aimed at providing a tool for stability analysis, performance evaluation and controller design.

Various aspects regarding structural and functional architecture, task complexity and flexibility, modeling issues and options, specific problems for sensing and commanding position have been presented.

Simplifications and approximations have been considered to provide a simple yet adequate design and analysis tool that would offer a good insight into the complex behavior of an uninhabited/autonomous airvehicle.

Preliminary numerical results show intuitively correct responses but future work is still needed to complete the entire simulation package and validate it.

11. REFERENCES

1. Rumbaugh et al., Object-Oriented Modeling and Design," Prentice Hall Inc. 1991
2. Kane T. R., Levison D. A., "Dynamics: Theory and Applications," McGraw-Hill Inc., 1985
3. Etkin B., "Turbulent Wind and Its Effect on Flight," Journal of Aircraft, No 5 1981
4. Perhinschi M. G., "A Model of Atmospheric Turbulence for Rotorcraft Simulation and Analysis of Stability and Performance," Proceedings of the AIAA Modeling and Simulation Technologies Conference, Boston MA, USA, August 1998, pp. 635-642
5. Beal T. R., "Digital Simulation of Atmospheric Turbulence for Dryden and von Karman Models," Journal of Guidance, Control and Dynamics, No. 1 1993
6. Anon., "Military Specification - Flying Qualities of Piloted Airplanes," MIL-F-8785C, 1985
7. Etkin B., "Dynamics of Atmospheric Flight," John Wiley & Sons, New York, 1972
8. Perhinschi M. G., Prasad J. V. R., "A Simulation Model of An Autonomous Helicopter," Proceedings of the RPV/UAV Systems Thirteenth Bristol International Conference and Exhibition, Bristol, United Kingdom, March 1998, pp. 36.1-36.13
9. Perhinschi M. G., "Parameter Optimization Via Genetic Algorithm of Fuzzy Controller for Autonomous Airvehicle," Proceedings of the AIAA Guidance, Navigation, and Control Conference, Portland OR, USA, August 1999, pp. 790-797

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37246

AIAA-2000-4284

MATHEMATICAL MODELS IN SUPPORT OF HARDWARE IN THE LOOP SIMULATIONS

Richard Gorecki*[1]
Defence Science and Technology Organisation,
Salisbury, South Australia, 5108

Abstract

The DSTO Hardware in the Loop (HIL) facility, is multi-purpose and also under continuous development at this time hence restricting its availability to each project utilising it. The main purpose of the facility is to conduct HIL simulations using missile hardware. Two complementary mathematical models have been written in order to increase its utility and to speed up the development of missile simulation models. One is a model for simulating the facility and the other is a baseline 6 DOF model of a generic missile. These models have been combined in a digital simulation model of a missile in a HIL facility.

1. INTRODUCTION

The idea for simulating the HIL facility was put forward by M. E. Sisle in a private communication who said it was the practice in the Raytheon Company. A simulation model of this type is very useful in studying the influence of the facility hardware limitations on experimental outcomes and to work out error bounds on these outcomes. A possible example of this can arise with the need to mount RF sensors away from the centre of rotation of the motion table in order to be clear of metal structures. This introduces parallax effects into the HIL simulation, which would need to be quantified. Another use for such a model is the remote integration of the missile model with the facility hardware.

Weapons Systems Division (WSD) of DSTO acquires and produces missile simulation models for its own studies and is the main supplier of models to other establishments in the defence community. There are limited manpower resources to do this so the Division is looking at ways of speeding up the development of simulation models. One method being investigated here in collaboration with other countries is the reuse and combination of previously developed and appropriate missile sub-system models using computer inter-operability software to allow programs to inter-operate across language barriers. A complementary method that is now being used is to develop missile 6 DOF simulation models from a baseline template model of a generic missile engagement.
This paper presents an overview of the mathematical models focussing on the underlying assumptions. More complete descriptions are given in references 1 and 2.

2. MATHEMATICAL MODEL OF A HIL FACILITY

This is a baseline model that essentially solves the instantaneous geometry between the missile unit detector and the position of the source representing the target in the scene generator. The model has a representation of the 3 axis motion table dynamics including data transmission delays. At this stage the motion only of the virtual source on the Radio Frequency Scene Generator (RFSG) horn array is modelled with the scene generator response being represented by transmission delays. The main model output is the position of the virtual source with respect to the detector bore-sight. In order to run closed loop simulations the model needs to be combined with a 6 DOF missile model, which is essentially driven by the bore-sight errors.

The RF environment is not modelled so that the bore-sight errors are geometric. The model does not address the influence of the ethernet, A-D and D-A conversion or sampling intervals on the simulation.

2.1 Motion Table

The motion table currently in this facility consists of a 3 axis table whose payload is the unit under test and a 2 axis table whose payload is an IR radiation source representing the target.

The table model requires transfer functions for each gimbal, which relate the motion table responses to demanded angles and rates. At this stage simple lag responses are assumed. To enable the proper modelling of the motion table rate and position limits the table gimbal position responses are modelled as unity feedback loops.

[1] *Senior Research Scientist

Copyright © 2000 by R. Gorecki. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

The table demands are derived from the 6 DOF airframe model rotation rate relative to the variable window frame[3,4,5] and resolved into payload axes. The table actual rotation rates are used as inputs to the missile tracker model of the space stabilisation loop. The implementation is shown in figure 1, which also summarises how the variable window frame is implemented.

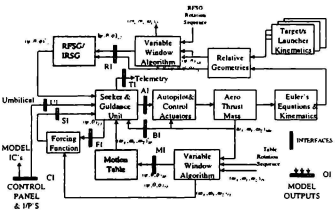


Figure 1. Block diagram of HIL system model with emphasis on the hardware integration

2.1 Radio Frequency Scene Generator

The horn array is on a spherical surface and the RF output of a triad of horns is combined to produce an electric field at the focus of the array whose phase front is tangential to the required direction of arrival. This is equivalent to creating a virtual source on the array surface at the point where the normal to the phase front intersects the surface. The motion table is located so that the origin of the HIL reference axes, ie the centre of rotation of the table gimbals, coincides with the array focus. Hence we can go from spherical coordinate to Cartesian coordinates and thus get the Euler angles of the normal to the phase-front. The rotation about the X axis is implemented by the polarisation of the field.

This model assumes that the algorithms and the control of the powers and phases of the triad of horns places the virtual source in the correct position to give the demanded angle of arrival at the focus of the array. The dynamics of this process is modelled by a delay, which represents the transmission delay.

2.2 Parallax and Parallax Correction

If the antenna is displaced from the array focus then parallax is introduced if the angle of arrival is calculated for the array focus. The displacements may be between the antenna and gimbal, gimbal and motion table centre of rotation or HIL reference origin or array focus and HIL origin. Parallax can be corrected by calculating the angle of arrival demands at the focus of the array that would give the required angle of arrival demands at the antenna position. Using the concept of the virtual source and assuming for the moment that the apparent radiation from this point is isotropic then the correction is calculated using the following procedure and solid analytic geometry.

Assume that the radiation at the antenna has the correct angle of arrival then the virtual source on the array surface is on the line passing through the centre of the antenna and having the same direction numbers as the LOS. The direction numbers of the line from this position to the array focus can then be used to define the corrected angle of arrival at the focus

2.3 HIL Geometry

The actual LOS rate orthogonal components in the detector frame are required to generate idealised guidance commands. These can be derived from the actual Euler angle rates and angles of the LOS from the antenna to the virtual source. It is best to do this relative to the earth frame since by definition there is no rotation about the X_{LE} axis and hence the Euler rates and magnitudes can be calculated using relative velocities and positions. Thus the relative velocity and position components of the virtual source with respect to the antenna are calculated in the HIL reference frame and then transformed into the earth frame.

2.4 Unit Hardware Geometry

The position components, in the unit antenna axes, of the virtual source with respect to the antenna are used to calculate the actual geometric bore-sight errors. These are inputs to the missile seeker model and hence drive the homing loop.

2.4 Variable Window and Forcing Functions

The variable window algorithm adjusts angle of arrival and motion table demands so that the LOS attitude relative to the seeker reference frame is independent of the relationship of the variable window frame with the software space reference frame (missile body centred, Earth).

If a variable window frame is used, the angle of arrival demands and the motion table demands are adjusted so that the LOS attitude ie the angle of arrival of the radiation relative to the seeker reference is not changed. Since the responses of the motion table and the scene generator are not the same then an error will be introduced. Also the table and scene generator motion will be shifted to different parts of their range so if their responses are non-linear then errors will be introduced. For example if the variable window frame is tied to the LOS frame then the scene generator motion will be restricted about its zero position while the table motion will be little affected

except near intercept. (Since missile rates are much greater than LOS rates.)

Forcing functions also need to be injected into the antenna to cancel out the variable window frame rotation. The forcing functions are components of the variable frame rotation rate resolved into antenna axes through the motion table angles and then the seeker head angles. So again, errors will be introduced due to the dynamics of the table and to noise and latencies in the head angle actual values.

3. BASELINE TEMPLATE MODEL OF A GENERIC MISSILE

There are many common elements among models. For example 5 and 6 DOF models of most weapons use the rigid body equations for the airframe model. They also use similar models for the thrust, mass and balance properties. Also in common are target maneuver models and kinematics, missile kinematics, relative geometry and LOS calculations, transformations between axis systems, Euler rates, quaternion rates and atmosphere models

The parts of the models specific to individual weapons are the aerodynamic data, the thrust, mass and balance data and missile sub-systems that may be modelled. These include tracker dynamics, sensor gain pattern, target signal measurement and processing, the guidance unit, target signatures, autopilots, actuator dynamics and instruments. These groupings of the common and specific elements are shown in figure 2.

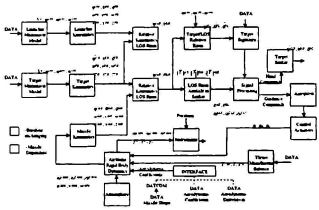


Figure 2. Block diagram of generic baseline model

Computer Simulation Models of many new missile systems will be required in the near future. In order to meet this requirement within the time and effort constraints, a baseline mathematical model of a typical generic missile reflecting in service missiles has been written. The model includes idealised representations of most of the specific missile sub-systems referred to above. The idealised sub-models should be easily replaceable with truer sub-models to develop a simulation model of an existing weapon. The intention is to build a library of sub-system models so that the most common missile types can be represented.

The baseline model uses, as much as possible widely used standards for nomenclature and axis system conventions. The main source document for these is reference 6 from which much of the notation is taken and there are also ideas taken from reference 7. As well as this, there is a distillation of information obtained from numerous books on aerodynamics and reports on missile models and simulation programs.

Computer simulation models based on this model have been programmed. The programs are modular with a view to reusable code and the application of object oriented programming techniques providing the readability of the program is not muddied. The baseline programs use always-stable Runge Kutta numerical integration algorithms which can be changed once a derived simulation model has been verified and validated. The changed model will then need to be re-validated.

3.1 Common Elements of Model

One of the rules that is used to determine the simulation program structure is to separate the code that models sub-systems from the housekeeping type code that models the environment or relates the interaction of one subsystem with another.

For example a missile seeker tracking a target involves many sub-systems coupled together physically and interacting with each other. Thus there is the target with a certain orientation and position in space emitting or reflecting radiation. A detector mounted in gimbals attached to the missile body receives this and measures bore-sight errors, which are used to generate control signals for tracking and missile control. The control motivators are attached to the body probably with some other orientation and cause the missile to maneuver so as to intercept the target. Equations describing the dynamics of all these systems in relation to the same frame of reference can be very complex and difficult to derive. It is far easier to write equations of each sub-system model in relation to its own frame of reference. Then transformation matrices facilitate the interaction between sub-system models.

So in the above example, the relative geometry, transformations between frames of reference and the generation of motivating forces are not part of the sub-system models and should be separate from them. Note some of these processes are performed by missile sub-systems eg resolvers in guidance units,

algorithms used in computers in non-linear autopilots and as such should be part of the sub-system model.

3.1.1 Housekeeping Aspects

Basic functions in the model include the generation of Euler and quaternion rates and transformation matrices and their operation on vectors. Basic modules include the atmosphere model, the kinematics and relative geometry of the various bodies and objects being modelled. These are the attacker/illuminator, the missile/detector and targets that may include multipath images, clutter patches, flares etc. As well as this would be the RF or IR environment.

3.1.2 Subsystem Models

These include maneuvering target models and target aspect and signature models. Missile subsystem models of the motor, the airframe mass, inertia and balance properties and airframe dynamics would be the same over the range of missiles of interest. For example using the rigid body equations to model the airframe dynamics would be sufficient for the applications intended for any derived models. Also the airframe mass, inertia and balance model will depend on the type of motor eg core burning or end burning rocket motors or turbo jet motors. Generic data driven models can be written for each type.

3.2 Weapon Specific Generic Elements

3.2.1 Seeker and Guidance Unit

The Guidance section or target seeker includes the sensor hardware, sensor positioning circuits and signal processing circuits.

The generic model of the guidance section is limited to the stabilisation loop of the antenna positioning circuits, the head tracking loop and the calculation of the guidance commands. At this stage the target is assumed to be a point source so the error signals that drive the tracking loop are the geometric bore-sight errors in each channel. Some trackers also use estimates of the components of LOS rate in antenna gimbal axes. In this case the geometric LOS rate is used. These LOS rate components are used in the calculation of the PN guidance demands.

At this stage only space stabilised trackers are considered. These trackers compensate for body motion and in the absence of error signals will remain pointing in the same direction in space. If the error signals are the bore-sight errors then the tracker will try to align itself with the measured LOS and in the steady state the bore-sight rate will equal the LOS rate. This result can be used in the calculation of PN guidance commands. The schemes that are commonly employed to isolate the detector from the body use two or more gimbals arranged so that each gimbal is aligned with an Euler rotation axis. A free gyro is used in some IR missiles with the gyro spin axis aligned with the bore-sight. This is modelled as a two-gimbal tracker with the gimbals aligned with the control motivators.

Tracker configurations that are modelled include nodding trackers, twist and steer type trackers and compound, nodding plus free gyro trackers.

The trackers are modelled as unity feedback loops with limiting on gimbal rates and position relative to the missile body.

The guidance unit model implements the simplest form of the PN navigation law, which is generally used in air-to-air missiles. Other guidance laws that will be implemented are augmented PN for air-to-air missiles, pursuit and lead angle for surface to surface missiles and command guidance and beam riding for surface to air missiles. There will also be a requirement to model mid course guidance demands from a third platform (normally the launcher).

First the PN guidance command components are calculated in the detector frame from estimates of the LOS rate components in antenna axes and the estimated closing velocity. Then components are calculated in the autopilot frame which when transformed to the detector frame equal the required PN guidance components.

3.2.2 Autopilots

The type of autopilot modelled is the standard proportional autopilot with airframe acceleration feedback as the main feedback controlling the loop gain and body rate feedback controlling the damping. The autopilot model does not include phase compensation and is a low frequency model in that body-bending filters are not included and the servo response characteristic is first order rather than second order. Usually the servo gain and the aerodynamic response are predetermined so the feedback gains are designed to give the desired closed loop response over as much of the full flight envelope as possible. This can be better achieved by using gain scheduling.

One of the aims of the template model was the ability to rapidly construct a missile simulation model that can simulate stable closed loop homing. Airframe characteristics such as mass and balance can fairly easily be measured or estimated and aerodynamic data can be derived from the shape (eg using MISSILE DATCOM software) and an airframe model can be quickly (relatively speaking) verified. However the configuration of the autopilots and guidance and tracking unit may not be as readily available. So this standard form was chosen, and to

complement it, the equations to dynamically calculate the feedback gains to give the chosen closed loop response over the full flight envelope. Thus the autopilot behaves in an ideal way always being matched to the airframe and the dynamic pressure.

The autopilots may include gravity compensation if the missile is roll stabilised. The models for the lateral autopilots are identical. The main inputs to the gain calculation are the airframe force and moment semi non-dimensional stability derivatives, mass and moments of inertia and the required natural frequency and damping. The latter two should be optimised to the airframe characteristics.

At this stage there has been no work done on a roll autopilot tuned to the airframe and dynamic pressure. This will probably use proportional plus integral plus derivative feedback controlling the roll angle. Note the base airframe model does not include induced rolling moments due to protuberances, due to wing or fin alignment errors or due to simultaneous and unequal pitch and yaw incidence. The airframe also has two planes of symmetry thus there are no inertial cross-coupling effects in roll. So the roll autopilot model in the generic template only needs to generate motivator demands for the missile initial roll command.

3.2.3 Autopilot Instruments

The missile autopilot accelerometers and rate gyros are modelled as having unity gain with measurement limits. The dynamic characteristics of these instruments are not modelled here but they are generally represented by a second order transfer function with a wide bandwidth compared to missile airframe response. Notch filters for vibration isolation are sometimes included. These would be included in high fidelity models for design purposes but not in low frequency models.

The generic models simply calculate the accelerations and rates that would be sensed by the instruments and limits these to typical values. The accelerometers may be displaced at some distance from the centre of gravity so they will also sense a tangential acceleration due to rotational acceleration. In this model they are assumed to be at the critical position.

3.2.4 Servos for Control Motivators

Servos in air-to-air and surface-to-air missiles are high bandwidth devices having a response in phase, equivalent to a second order system with natural frequency and damping greater than 150 r/s and 0.5 respectively. This is significantly higher than missile airframe frequencies and thus servo models in low frequency 6 DOF models tend to be low fidelity models.

High fidelity models of pneumatic or hydraulic servos would model fluid compliance and pressure rise, piston movement, torques from actuators, aerodynamic hinge moments and coulomb friction, inertia effects and motivator acceleration, velocity and position with limiting of the latter two. The modelling of fluid compliance and pressure rise effects involve very high gains (inverse of the bulk modulus) and hence rates of change of pressure which require very small integration intervals for stable solution. So a lesser fidelity model allowing faster computation would assume incompressibility and only model the actuator dynamics.

Further simplification can be achieved by ignoring motivator acceleration so that the servo model is equivalent to a simple lag. The pressures acting on the pistons and how the forces are transmitted to the motivator determine the time constant. The maximum slew rate of the actuator is also determined by system pressures and maximum fluid flow rate that is determined by valves. If using this level of model then it is assumed that the servos can produce enough torque to overcome torques arising from aerodynamic hinge moments and coulomb friction.

This model assumes four motivators arranged in two pairs controlling motion in the missile pitch and yaw planes. Roll control is achieved by superimposing differential deflection demands on the demands of either one pair, or both pairs of motivators.

The servos are modelled as unity feedback loops with limiting on servo rates and position.

3.2.5 Airframe Configuration

The airframe dynamics are described by the equations of motion for a rigid body. This is one of the common elements of the model. The airframe configuration, however, is weapon specific, and this is expressed in the aerodynamic data that is used in the calculation of the aerodynamic forces and moments, and the moments of inertia and products of inertia data in the equations of rotational motion.

The equations indicate that there can be much cross coupling of motion between planes particularly if rolling motion is present. So the generic baseline assumed is a cruciform missile having two planes of symmetry ie no cross products of inertia, and using some form of Cartesian control and also roll control. This minimises the data required to model the airframe characteristics.

The aerodynamic forces and moments that are input to the airframe dynamical equations use coefficients that reflect the airframe configuration.

Reference 8 shows that the contributions to each coefficient can be related to the instantaneous dynamical state specified by the translational and rotational velocities and accelerations, for example the set $(u, v, w, p, q, r, \dot{v}, \dot{w}, \dot{p}, \dot{q}, \dot{r})$. The coefficients are made up of component parts that are the partial derivatives of the coefficient with respect to non-dimensional forms of each of the state specifiers. These are called stability derivatives and are measured or calculated with respect to stability axes that may or may not be aligned with the body axes. If they are not aligned then the derivatives must be transformed into the body frame. The derivatives relating to velocity components are called resistance stability derivatives, those related to angular velocities are the rotation stability derivatives and those relating to acceleration are the acceleration stability derivatives. There are 6 derivatives in relation to each state specifier used.

The generic model uses the following set of body referenced stability derivatives. The moment and damping derivatives about the three axes and the force derivatives along the three axes. It also uses derivatives relating to the contribution of motivator deflections to the body moments and axial forces.

Other useful data are the aerodynamic hinge moment derivatives with respect to motivator deflection and incidence. These may be required in models of certain servos eg torque balance systems.

Note that the rotational derivatives are normally given with respect to some reference point, usually the centre of gravity position for either the motor unburnt or burnt condition. Thus the rotational derivatives need to account for the shift in the centre of gravity as the motor burns. There are good derivations of these in reference 9. This reference also gives explanations of the physical meaning of the more important stability derivatives.

3.2.5 Airframe Mass, Balance & Moments of Inertia

The model of the behaviour of these airframe parameters is generic but the input data reflects characteristics of the behaviour.

The mass, inertia and balance properties vary with time as the motor burns. They decrease from their initial values at the start of burn to their values at the start of glide. It is assumed that these quantities vary linearly with the impulse. This holds if the propellant burns uniformly outwards from the core along the length of the motor. The variation is non-linear if the propellant burns from the end of motor but the deviation from linearity can be ignored in this context since the motor generally burns for a short time and the missile autopilot effectively reduces this effect.

At a given thrust level the propellant burn rate is constant so the mass varies linearly with impulse. There may be loss of mass from venting of gases from gas servos or hydraulic fluid from hydraulic servos but since this is much smaller than the loss of propellant mass it is usually ignored. The effect of this on moments of inertia and centre of gravity is also ignored.

If the missile to be modelled has jettisonable boosters or thrust vectoring equipment or uses some form of jet propulsion then it might be better to model the above properties by tabular data versus time.

4. SOME SIMULATION RESULTS

The two models have been combined in a computer program that simulates a generic missile in a HIL facility. The following example illustrates the effect of parallax on the verity of a simulation of an engagement.

In the engagement the missile was fired at a crossing target flying at a higher altitude. This set up, in antenna axes, an initial step of LOS azimuth rate of about 3 deg/s and of LOS elevation rate of about 0.5 deg/s.

Four cases were looked at in which the histories of seeker estimates of the LOS elevation and azimuth rate components as well as the bore-sight errors were compared. Also plotted were the software space LOS elevation and azimuth rate components, the forcing functions and the HIL LOS rate components, all in antenna axes. The HIL LOS rates are between the virtual source and the antenna in hardware space.

The base case (a) uses the missile digital simulation model. The remaining cases use the missile in a HIL facility model. A variable window tied to the LOS frame and forcing functions were used since the modelled RFSG is of limited size. In case (b) the antenna is at the HIL origin ie no parallax. In case(c) the antenna is projecting forward from the HIL origin so there is parallax. In case (d) the parallax is corrected.

Comparing results in figures 3 and 4 we see that the difference between case (a) and (b) is small except at intercept, where divergence occurs. In figure 4 there is a small HIL LOS rate (ideally zero) due to the lags and delays in the table gimbals and scene generator.

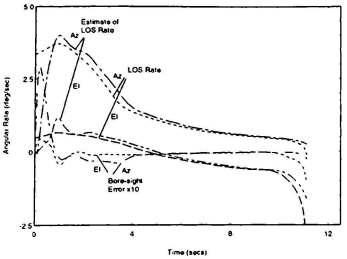


Figure 3. Case (a): LOS rate components in antenna axes.

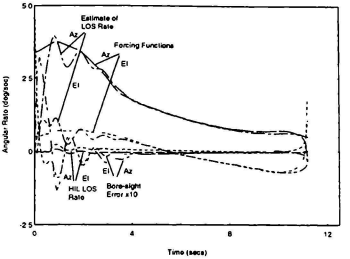


Figure 5. Case (b): LOS rate components in antenna axes: variable frame equals LOS frame. Parallax

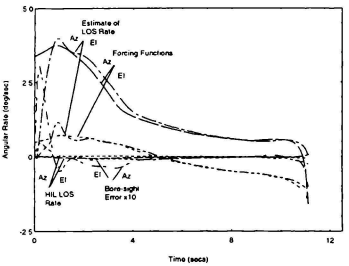


Figure 4. Case (b): LOS rate components in antenna axes: variable frame equals LOS frame. No parallax

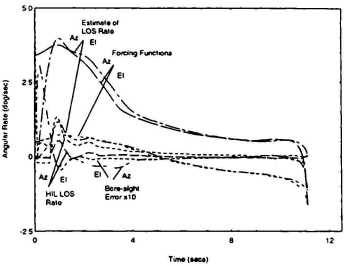


Figure 6. Case (b): LOS rate components in antenna axes: variable frame equals LOS frame. Parallax corrected

Comparing estimated LOS rates in figures 4 and 5 we see that parallax introduces a significant oscillation in the early part of the engagement. At this time missile rotation rates are larger since it is responding to the step in LOS rate. As the missiles settles into the PN course the oscillations decay and an intercept occurs. However the simulation results are not valid.

In certain conditions of missile guidance time constant and antenna projection and HIL facility dynamics these oscillations may increase and an unstable situation would prevail.

When parallax correction is applied, as in figure 6, the oscillations are eliminated and the output variable histories are the same as in figure 4. Note the differences in the HIL LOS rate histories (figs 4,5 &6) as parallax occurs and then is corrected.

6. CONCLUSIONS

A framework for a HIL system model has been developed. This mainly involves fairly complex transformations between the axis systems used in the various pieces of hardware that comprise the HIL simulation. The intention is to add higher fidelity models of the motion table dynamics and equations that calculate the electromagnetic field at the antenna.

The template model is currently able to model Air-to Air missiles. It is the intention to expand the range of subsystem models to include other types such as Surface-to Air and Sea Skimming missiles. As well as this it is planned to expand the scope of the model to include more of the mission level aspects.

Generic models of the RF environment will be added together with representations of antenna gain patterns and RF signal processing. This will match the expanded capabilities of the HIL facility model.

6. REFERENCES

1. Gorecki R.M., "A Mathematical Model for Simulating a Hardware in the Loop Facility". DSTO-TR-0931, March 2000.
2. Gorecki R.M., "A Baseline 6 DOF Mathematical Model of a Generic Missile". DSTO-TR-0930, March 2000.
3. Lange A.S."Semi-Physical Simulation of Guided Missiles", Comput. & Elect. Engng, Vol.1 pp. 119-142 Pergamon Press, 1973.
4. Snyder H.P.,"Artificially Generating Line-of-Sight Rates For Hardware-in-the-Loop, Missile Interceptor Simulations" Proceedings of the 19?? Summer Simulation Conference
5. Gorecki R.M., "Variable Window and Forcing Function Algorithms for Hardware in the Loop Simulations". DSTO-TR-0932, March 2000
6. American Institute of Aeronautics and Astronautics.," Recommended Practice – Atmospheric and Space Flight Vehicle Coordinate Systems", ANSI/AIAA R-004-1992
7. P. Garnell & D. J. East., "Guided Weapon Control Systems",1st Edition Pergamon Press 1977
8. Jack N. Nielsen., "Missile Aerodynamics" Nielsen Engineering and Research, Inc 1988.
9. John H. Blakelock., "Automatic Control of Aircraft and Missiles" 2nd Edition John Wiley & Sons 1991

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37247

AIAA-2000-4285

USING SIMULATION TO OPTIMIZE SKI JUMP RAMP PROFILES FOR STOVL AIRCRAFT

Gregory Imhof* and William Schork
Naval Air Warfare Center/Aircraft Division, Patuxent River, MD

Abstract

Ski jump ramps are used aboard the Naval ships of several countries to reduce AV-8B Harrier takeoff run distance and wind-over-deck requirements, as well as to increase aircraft takeoff gross weight capability over a flat deck carrier. As part of the Joint Strike Fighter program, a simulation effort is underway to evaluate various ramp profiles, ramp optimization methologies, and minimum launch airspeed criteria. Results of these evaluations could be used to provide an advanced STOVL aircraft with greatly improved takeoff performance using ski jump ramps, while not becoming a design driver for landing gear or ship designs. This paper describes preliminary results found, as well as planned work and the benefit to the JSF program.

Abbreviations/Acronyms

CASTLE	Controls Analysis and Simulation Test Loop Environment
GLG	Generic Landing Gear
JSF	Joint Strike Fighter
MLA	Minimum Launch Airspeed
RASTO	Ramp Assisted Short Takeoff
STO	Short Takeoff
STOVL	Short Takeoff and Vertical Landing
WOD	Wind Over Deck

Introduction

Compared to current aircraft, future STOVL aircraft will have significantly improved flight control systems, which will be able to offload many of the pilot tasks for takeoff and landing. By taking the pilot control variability out of a STO, it may be possible to change not only the criteria used to determine MLA, but also the amount of excess airspeed added on as a safety margin. This translates to shorter takeoff distances and/or increased takeoff weights as compared to current methods. The effort described in this paper investigates ramp designs and launch criteria to determine potential benefits for future Joint Strike Fighter STO operations. Models developed for this project could be used for future JSF work.

* Member, AIAA
This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

Purpose

Although the US Navy has considerable operational and test experience with ski jump ramps, there has been little research performed in the past with regard to:

1. Optimal ramp profiles for maximum takeoff performance
2. Effect of various ramp profiles on aircraft landing gear and structural loads, including fatigue and other life-cycle issues
3. Minimum launch airspeed criteria specifically for RASTO operations
4. Impact of various ramp designs on the ship itself

The first three of these items are being investigated in this effort, while the fourth is a part of a separate study. This paper focuses primarily on work completed for the second item, and discusses upcoming work on the others.

Background

US Navy/Marine experience

The Boeing AV-8B Harrier is a true STOVL aircraft, in that it routinely performs short takeoffs and vertical landings. The United States operates its Harriers from amphibious assault ships (LHA/LHD class), which are not equipped with catapults, ski jump ramps, or arresting gear and are considerably smaller than CV/CVN class large deck aircraft carriers. The unique STO and hover capabilities are obtained through a group of variable angle nozzles for vertical lift and a reaction control system for stability and control, which uses engine bleed air to provide thrust through several small nozzles located on the aircraft.

Since none of the LHA/LHD class ships currently have ramps, Harrier takeoffs require using most of the deck length for takeoff at high gross weights. There are many advantages to reducing the deck run required, including being able to operate from smaller ships or flying concurrent flight operations.

One way to reduce the deck run required is to install a ski jump ramp at the bow of the ship. The US Navy has conducted many shore and ship-based tests with smooth and segmented ramps over the years to demonstrate the performance advantages of a ramp assisted takeoff.

Besides the Harrier, other aircraft tested on ski jump ramps include the F-14, F-18, and T-2. These conventional carrier based aircraft can benefit from ski jump ramp takeoffs, even with relatively small 2-3° exit angle ramps installed at the ends of the catapults.[1]

Foreign Navy Experience

Several foreign navies have operational experience with ski jump equipped ships, including the Italian, Spanish, and British. The majority of these ships are equipped with 12° ramps, while the others have 6° ramps. The NAWC/AD's Carrier Suitability Division has conducted many flight tests from foreign navy ships to establish the AV-8B shipboard operating bulletins. RASTO's and vertical landings are conducted to determine wind over deck requirements at various takeoff weights. Instrumented aircraft are used for these tests, and some of the data collected has been used by this project for Harrier simulation validation and comparison purposes.

Tests were conducted in the 1980's from the Spanish Ship PRINCIPE DE ASTURIAS, which has a 12° ski jump ramp. From these tests, it was determined that WOD requirements could be 30 knots lower, deck run 350 feet less, or maximum payload increased up to 53% as compared with a flat deck STO. For example, with the same WOD and payload as a flat deck STO, the deck run for the RASTO would be 350 feet shorter. This increased takeoff performance provides many benefits to the operations of the aircraft and the ship:

1. Increased takeoff weight (3,000 - 5,900 pounds) allows additional stores weight for weapons or fuel, even in tropical day conditions.
2. The ship has more maneuvering flexibility during flight operations, since it is less dependent on ship speed to generate the high WOD required for launch from a flat deck ship.
3. Reduced deck run frees up a significant portion of the flight deck, allowing concurrent operations with helicopters and other vehicles on the aft section of the ship.
4. Pilot workload is reduced as compared to a flat deck STO. The aircraft can be flown "hands off" for a few seconds after ramp exit, allowing the pilot more time for instrument scans and engine monitoring.[2]

Similar results were obtained for other 12° ramp-equipped ships. The Italian ship GIUSEPPE GARIBALDI has a 6° ramp, and while the takeoff performance improvement was not as large when compared to a 12° ramp, there are still significant improvements over a flat deck ship takeoff.[3]

Typical RASTO Flight Path

The flight path trajectory from a ski jump takeoff has three main parts, as shown in figure 1. The first section is the deck run on the ship, up to ramp exit. Deck speed loss during ramp transit is negligible. At the ramp exit, the aircraft has a positive rate of climb due to the ramp induced vertical velocity. This rate of climb will continue to decrease, since the aircraft does not yet have enough airspeed for the wings to create sufficient lift (normal acceleration less than 1g). If the rate of climb becomes negative, the aircraft will begin to loose altitude. The highest altitude achieved in this scenario is called the apogee. As the aircraft continues to accelerate, the aircraft passes the inflection point (normal acceleration equal to 1g), as it can now generate sufficient aerodynamic lift and the rate of climb will begin to increase.

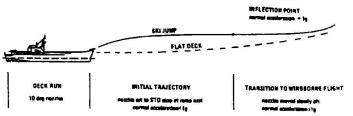


Figure 1: Typical RASTO Flight Path

Segmented vs. Smooth Ramps

All operational ramps today have a smooth profile, but tests have been conducted in the past which show that segmented ramps can be used instead. A retractable segmented ramp is an ideal implementation for operational use. By retracting the ramp into the deck, greater area for spotting aircraft would be available when the ramp is not is use. Depending on the design, different ramp profiles could be set for different aircraft, with each profile being the optimum for the particular aircraft. A segmented ramp is typically simpler to design and build, since it is a collection of flat plates rather than one continuous curved surface. There are disadvantages to segmented ramps, such as oscillating landing gear loads, which will be discussed later.

Description of Models Used

Three simulations were used for this work, each with different capabilities. The CASTLE AV-8B simulation is a six degree of freedom simulation, with both offline and pilot-in-the-loop capabilities, and a complete AV-8B flight control model. CASTLE is the simulation executive developed by NAWC/AD, and can be used on Windows 95/98/NT, DEC VMS, and SGI-UNIX platforms.[4]

The AUTOJUMP AV-8B and F-18 simulations numerically integrate the equations describing an aircraft's motion during a ski jump launch. A Runge-Kutta numerical integration technique is utilized exercising a 5 degree of freedom model: horizontal and vertical translation, pitch rotation, and nose and main gear stroking. AUTOJUMP is an offline simulation with a simplified flight control model.

Only the CASTLE AV-8B simulation will be used for the minimum launch airspeed evaluations, while both AV-8B simulations will be used for the ramp optimization work, and all three were used for the segmented ramp study.

AUTOJUMP had been used in the past for other landing gear analyses, and the users were familiar with the structure and simplicity. In addition to its use with the ramp profile optimization work, AUTOJUMP was used during the validation process of the CASTLE AV-8B simulation, which had not been used for landing gear analysis before.

Aerodynamic and Propulsion Models

The 100% LERX aerodynamic and RR-402-408 engine/propulsion models are based on the McDonnell-Douglas AV-8B HIAOA model obtained in 1998. The 65% LERX aerodynamic and RR-402-406 engine/propulsion models are based on the USMC AV-8B trainer device 2F150A. Both models have known deficiencies, primarily in the ground effect and semi-jetborne flight regimes. The configuration being used for our simulation work is the 100% LERX aerodynamic and RR-402-408 engine configuration. Bihrle Applied Research has been tasked to update this model with emphasis on correcting the known deficiencies mentioned above. The AUTOJUMP and CASTLE simulations use the same aerodynamic and engine/propulsion models.

Landing Gear Models

The CASTLE Generic Landing Gear (GLG) model was developed for use as a baseline gear model for various fixed and rotary wing aircraft simulations. It is a six degree of freedom model that was designed to accommodate up to five struts. The generic design of the model allows the user to input the basic geometry of the landing gear, along with the strut and tire properties, to quickly generate a reasonable fidelity gear model. Additional detail can be added to the model as necessary for each airframe. The model is currently used by the F/A-18, F-14, S-3, C-2/E-2, X-31, and V-22 simulations.

The AUTOJUMP gear stroke model simulates a mass damper system with a single point of contact tire model. It is similar to the CASTLE GLG model in function.

Several modifications to the AUTOJUMP and GLG were required prior to use. The AV-8B specific landing gear loading vs. displacement curves and damping characteristics of the struts and tires were needed. A ramp routine was also required, which could support various ramps with easy modifications to the profiles. Initially, all of the operational shipboard ramps were modeled.

It was quickly realized that the parameters used for the nose gear strut displacement vs. loading and the damping parameters were not accurate. It was assumed that the shape of the curve was correct, but that it needed to be modified slightly. Strut deflections with the aircraft in static conditions and known weight/cg gave points to better anchor the design report curve. The results are shown in figure 2. Strut damping characteristics were estimated based on similar aircraft and by comparing simulation strut deflection time histories with flight data.

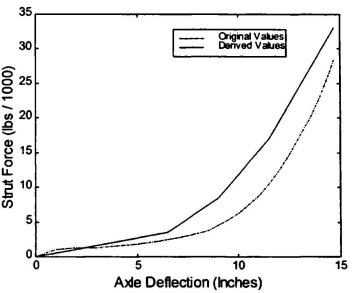


Figure 2: Nose Strut Load vs. Deflection Values

Pilot Models

Pilot models were needed for the CASTLE and AUTOJUMP simulations which could be used for the unpiloted offline runs. The CASTLE model is very basic, performing the following steps for a standard RASTO maneuver.

1. With brakes applied, set takeoff trim, STO nozzle stop angle, and nozzles to 10° down.
2. Run engine to full power, and release brakes when aircraft begins to roll. Maintain lineup during deck run.

3. When the pilot crosses the nozzle rotation line painted on the deck (approximately 10 feet prior to ramp exit), rotate the nozzles to the STO stop.
4. Capture and maintain target angle of attack using longitudinal stick.
5. Transition to wing-borne flight after the flight path inflection point by slowly raising the nozzles.

The AUTOJUMP pilot model uses a path follower technique for pitch attitude capture. A curve is fitted as a function of time between the current aircraft pitch attitude and the desired one. The derivative of this curve is the desired pitch rate. During each time step in the Runge-Kutta integration, an iterative process is performed to determine the proper control surface position that will provide the desired pitch rate.

Comparison with GARIBALDI flight test

Validation of the complete models for RASTO maneuvers was conducted to ensure that the models were representative of flight test time histories. Primary parameters compared were strut deflections, airspeed, pitch attitude, and angle of attack. Good correlation between flight data and simulation was found, and differences were attributed to dynamics of the landing gear and tires that were not completely modeled. Improvements to the tire model are planned and should add a significant improvement to the model.

Segmented ramp study

The first major task was to compare ramp assisted takeoffs with a segmented ramp versus a smooth profile ramp, and how a segmented ramp could be used with the existing structural and operational requirements of the aircraft. The initial focus was on the maximum angle change between segments that the aircraft and aircrew could tolerate. The results of this work were passed along to the separate work being conducted on shipboard ramp design and integration issues.

Previous flight tests did not have sufficient instrumentation to measure gear or store loads and accelerations. Therefore, it was only possible to compare trends and known limits instead of actual gear strokes and structural loads. For structural reasons, it was stated that the landing gear shall not compress to full closure at any point during the takeoff. Flight tests have been conducted to within one-half inch of full closure with no adverse results. This was set as the working limit of our optimization work.

Two types of aircraft with two different types of landing gear configurations were analyzed for several reasons. Previous F-18 ski jump test data includes landing gear loads and accelerations, which none of the AV-8B tests have. This was helpful in the validation of the landing gear models. There was also some concern that the bicycle arrangement of the AV-8B would be too different from the tricycle configuration of the JSF designs, and that the AV-8B based results would not be applicable to the JSF configurations. The F-18 model was used to demonstrate this compatibility.

The HMS Invincible and the 15 foot segmented ramp profiles are shown in figure 3. The smooth rounddown at the end of the Invincible ramp (the section past 140 feet) was added to all of the segmented ramps for consistency. The average segment angles for each profile are presented in table 1. Figure 4 is an exploded view of figure 3, and shows that even the 15 foot segments do not change the overall shape of the ramp significantly.

Table 1: Segmented Ramp Angle Comparison

Ramp Segment Length	Angle Between Segments
Seamless	0.1
5 ft	0.4
10 ft	0.9
15 ft	1.3

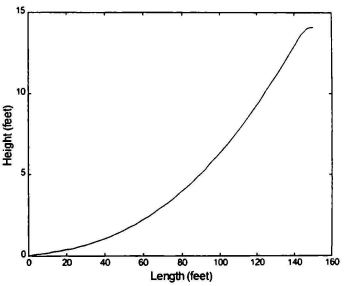


Figure 3: Invincible Ramp Profile

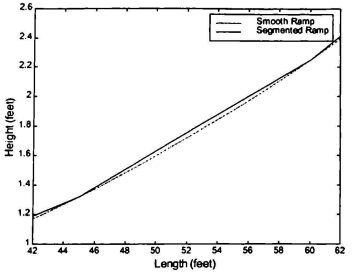


Figure 4: Invincible and Segmented Ramp Profiles

For the F/A-18 runs, a takeoff weight of 37,000 pounds and ramp entry speed of 113.5 KTAS were used. During the 1985 NAVAIRTESTCEN F/A-18 Ski jump Evaluation Tests, this was the heaviest and fastest case demonstrated. The AV-8B had a takeoff weight of 27,110 pounds with a ramp entry speed of 80 KTAS with a 30 KTAS head wind. This condition is typical for a ski jump launch of this aircraft.

Segmented Ramp Study Results

Time histories comparing the nose and main gear forces and strokes for the AV-8B on the 10 and 15 foot segment ramps are presented in figures 5 through 8. Time 0.0 is when the nose gear tire enters the ramp. At each ramp segment junction, the tire force responds to the change in ramp angle by compressing and increasing tire force. This compression is oscillatory and is a function of the ramp segment length and angle. When comparing between the segmented ramp profiles, as the ramp segments get longer, the tire force oscillation amplitudes increase and frequencies decrease. It was found that the aircraft flyaway pitch attitude, and therefore takeoff performance, is unaffected for these different ramp segment lengths. For the conditions analyzed, ramp angle segments up to 1.3° did not produce any adverse effects on the gear loading or the vehicle ramp flyaway performance.

A limited structural analysis showed that for a clean aircraft (with no external stores), the structural loads during a RASTO would be within the vehicle's structural limits. The structural engineers had concerns of the possibility of the tire force oscillation exciting a resonant frequency of an aircraft system, such as the landing gear, and/or a local structural element. They also commented that the simplistic tire deflection model being used could conservatively predict higher amplitude gear forces. (The existing tire model over predicts tire deflections at the ramp segment junctions, and has no tire damping.) For an actual ramp takeoff with constant segment lengths, the tire force amplitudes may be less, but the oscillatory characteristics may still be present. This oscillation is a function of the ramp segment length and the aircraft ramp entrance speed. The final recommendation was to try to reduce the oscillations.

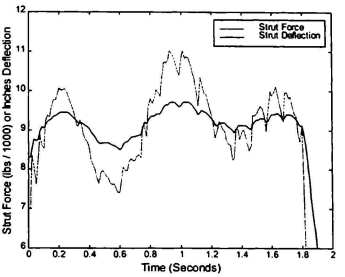


Figure 5: AV-8B Nose Gear Force and Stroke for 10 foot segmented 12° ramp

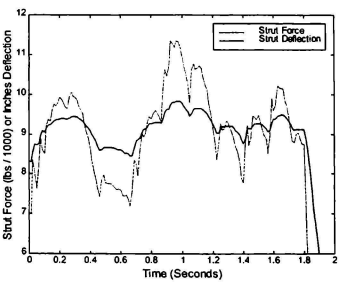


Figure 6: AV-8B Nose Gear Force and Stroke for 15 foot segmented 12° ramp

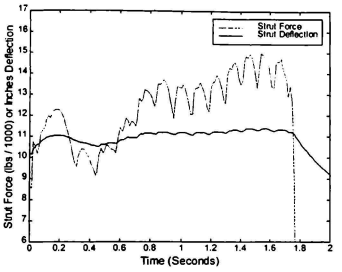


Figure 7: AV-8B Main Gear Force and Stroke for 10 foot segmented 12° ramp

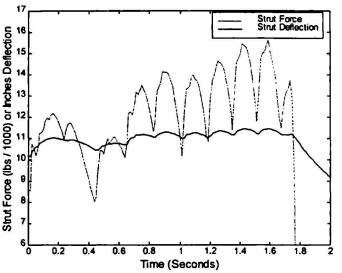


Figure 8: AV-8B Main Gear Force and Stroke for 15 foot segmented 12° ramp

One way to reduce the oscillation problem is to vary the segment lengths so that a constant frequency oscillation is difficult to achieve. Figure 9 presents a portion of such a varying segment length ramp, as compared to the smooth Invincible ramp. The segment lengths were chosen which maximized the segment's length and kept the angles between segments less than 1°. This is not the optimum design, but a test case to analyze a varying length segment ramp was deemed worthwhile.

Figures 10 and 11 present the nose and main tire forces and strokes for a varying angle segmented ramp takeoff. The tire force is still sensitive to changes in ramp angle, but the oscillatory characteristic is irregular and less likely to result in resonance-induced problems.

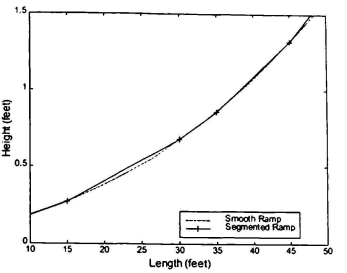


Figure 9: Section of the Varying Segment Ramp

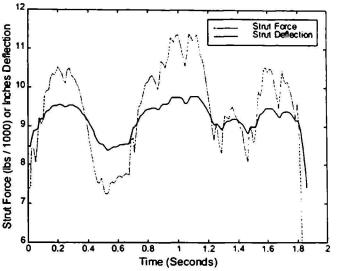


Figure 10: Nose Gear Force and Stroke for Varying Segment Ramp

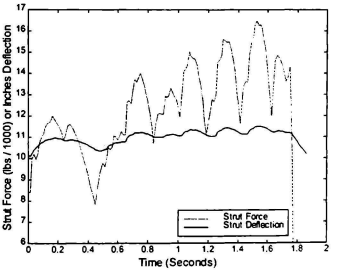


Figure 11: Main Gear Force and Stroke for Varying Segment Ramp

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37248

AIAA-2000-4286

A COMPARATIVE ASSESSMENT OF VARIOUS MULTIPATH MODELS FOR USE IN MISSILE SIMULATION STUDIES

Domenic Bucco* and Yiding Hu†
Defence Science and Technology Organisation,
Salisbury, South Australia, 5108

ABSTRACT

An important modeling consideration when assessing the performance of radar (RF) guided missile systems at low altitude over land or sea is signal multipath. In particular, for a typical missile simulation, the digital representation of multipath signals has been pursued using a variety of models that depend on the scattering characteristics of the surface. In this paper, we describe several approaches to multipath modelling and numerically compare them in the context of open loop studies of a generic semi-active RF guided missile against a low level target over the sea.

1. INTRODUCTION

An important modeling consideration when assessing the performance of radar (RF) guided missile systems at low altitude over land or sea is signal multipath. In fact, multipath can adversely affect the front receiver of active, passive and semi-active seekers as well as the rear receiver associated with semi-active guidance. Applications of naval interest include the performance of RF guided missiles against sea skimming targets as well as the tracking accuracy of onboard radar sensors. Models of multipath scattering phenomena that have appeared in the literature tend to follow a common theme in which the total signal is considered to be the combination of a direct signal and a reflected signal. However, within this theme, various methods for modeling multipath effects have apparently emerged. Three approaches to multipath modelling and simulation are assessed here in the context of performance studies of a generic semi-active RF guided missile.

One approach, specific to sea multipath returns, is based on the phenomenological vector model of Beard, Katz, and Spetner[1]. This approach, which we have designated as Model 1, makes use of empirical data derived from measurements conducted out at sea[2].

Another approach, which is more general, is based on the theoretical work of Beckmann and Spizzichino[3] regarding the scattering of electromagnetic waves from rough surfaces. In this approach, which was later modified by Barton[4], the specular and diffuse components of the signal multipath are characterized through the use of mean power reflection coefficients. A version of this approach based on the discretisation of the glistening surface into ten strips has been developed and is designated as Model 2. Here, Model 2 is very similar to that developed by Smith *et al*[5]. Note that in the above models, the total scattering signal is represented as the explicit addition of two components; a coherent (specular) component and an incoherent (diffuse) component, and these components are modelled independently. Other models in this category that have appeared in the literature include those described by Lopez[6], Osborne[7], Chisholm[8] and the software package TKMOD5C[9].

A third approach to predicting multipath effects on RF guided missile seekers is based on modelling the total scattering coefficient from the physics of the problem and this can be achieved using time domain or frequency domain techniques. Within this approach, we have derived three sub-models designated as Model 3, Model 4 and Model 5. Models 3 and 4 are time domain ground grid models while Model 5 is a frequency domain ground grid model and is based on the concept of iso-Doppler (isodop) contours. This latter approach is adapted from a technique generally used for predicting the effects of clutter returns on the seeker performance[10,11]. One major difference here is that the scattering coefficient central to the model is based on forward scattering of the RF waves rather than backward scattering as in the case of clutter returns[11].

These three approaches to multipath modelling will be assessed here in the context of open loop studies of a generic semi-active RF guided missile against a low

*Principal Research Scientist, Member AIAA
†Senior Research Scientist
Copyright © 2000 by D. Bucco & Y. Hu. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

level target over the sea. This paper extends the work produced earlier on multipath modelling[12]. Here, however, emphasis is placed on the numerical comparison of the three multipath modelling approaches.

First, a functional description of each model is outlined, with the underlying principles of the three methods being highlighted. The discussion is then followed by a quantitative comparison of the various features and limitations of each model. For the purposes of comparison, the performance of a generic RF guided missile against a low level target over the sea will be considered. A brief summary regarding the results of the quantitative comparison concludes the paper.

2. OUTLINE OF MULTIPATH MODELS

In this section, we briefly outline the main features of each of the multipath models that we shall compare in a later section. More details on each model can be found in the references.

2.1 Model Basic Structure

In order to simplify the description, the total signal received at the rear receiver of a semi-active missile system is used as an illustrative example of the solution process for each multipath model. Figure 1 shows the signal components at the rear receiver. These include a direct signal (D) and a scattering signal (S). The scattering signal can be further divided into specular/coherent (C) and diffuse/incoherent (I) parts. The specular point in this figure is defined as the reflection point from the smooth sea surface.

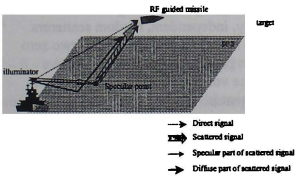


Figure 1. *The signal arriving at the rear receiver of a semi-active missile.*

The basic structure of the three approaches to multipath modelling can be readily described as a vector signal diagram as shown in Figure 2. The total signal (T) is formed by either

$$\vec{T} = \vec{D} + \vec{S} \quad (1)$$

or

$$\vec{T} = \vec{D} + \vec{C} + \vec{I} \quad (1')$$

and the major difference between each model is in the modelling of the scattered signal. The basic structure of the individual models is briefly summaried as follows:

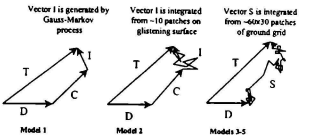


Figure 2. *Wave vector expression of the three approaches to multipath modelling.*

- **Model 1:** This model calculates specular and diffuse scattering signals. Wave field strength and phase of both C and I signals depend on the statistical fitting of experimental data. This is a time domain model.
- **Model 2:** This model calculates specular and diffuse scattering signals. The specular signal is computed as in model 1, but the diffuse signal is integrated from ten strips approximating the glistening surface. The scattering coefficient is simplified at low grazing angles[5]. This is also a time domain model.
- **Model 3:** This model integrates all scattering signal power from a ground grid with a log-scale mesh around the specular point. An analytical scattering coefficient model with the assumption of relatively rough surface[11] is used. This is a time domain model.
- **Model 4:** This model integrates all scattering signals from a ground grid in an extended glistening surface boundary. It has the same scattering coefficient model as model 3 above and is a time domain model.
- **Model 5:** This model integrates all scattered signals from a ground grid based on iso-Doppler (isodop) contours. Then the scattering signal is transferred from the frequency domain to the time domain for comparison. The model uses the same scattering coefficient model as model 3.

2.2 Direct Signal

The direct signal, *D*, is the same for all models. As an example, the voltage of the direct path for a 50Ω receiver system is

$$D=[100P_{Tx}G_{TxD}G_{RxD}\lambda^2/(4\pi R_{IM})^2]^{1/2}\, exp(-j2\pi R_{IM}/\lambda)\, exp(j\theta_D) \quad (2)$$

where P_{Tx} is the transmitted power, G_{TxD} and G_{RxD} are transmitter (Tx) and receiver (Rx) atenna gain patterns at the off-boresight angles of the direct path, $R_{IM}$the range from Tx to Rx, θ_D the Tx phase at the existing off-boresight angle, and λ the RF wavelength.

In equation (2) there is a fast phase shift due to missile motion Doppler phenomenon:

$$exp(-j2\pi R_{IM}/\lambda) \sim exp(-j2\pi f_{Doppler}\, t)$$

thus

$$f_{Doppler} = V_M \cos\theta / \lambda$$

where V_M is the missile speed and θ is the angle between vectors $\mathbf{V}_M$ and $\mathbf{R}_{IM}$. In order to not allow this well-known phenomenon to affect demonstrating multipath effects, we henceforth set all phase variations relative to the direct signal Doppler. Therefore, equation (2) becomes

$$D=[100P_{Tx}G_{TxD}G_{RxD}\lambda^2/(4\pi R_{IM})^2]^{1/2} exp(j\theta_D) \quad (2')$$

2.3 Specular Component

The specular signal, *C*, is calculated in the same way in models 1 and 2. For the 50Ω receiver, the voltage of specular or coherent signal component is

$$C = \{100P_{Tx}G_{TxC}G_{RxC}\lambda^2/[4\pi(R_{IS}+R_{SM})]^2\}^{1/2}\, \sigma_c\, exp(-j\alpha_c)$$
$$\sigma_c = \Gamma\rho_c\underline{D}$$
$$\alpha_c = 2\pi(R_{IS} + R_{SM} - R_{IM})/\lambda - \theta_c \quad (3)$$

where G_{TxC} and G_{RxC} are Tx and Rx antenna gain patterns at existing off-boresight angles, R_{IS} & R_{SM} denote the range from Tx to the specular point and from the specular point to Rx, respectively, θ_c is the Tx phase at the existing off-boresight angle of R_{IS}, $\underline{D}$ the divergence factor (using 4/3 earth's radius approach[11]), Γ the smooth planar (Fresnel) surface reflectivity for vertical polarisation (can be modified to other polarisation depending on Tx and Rx configuration)[3,11], and ρ_c the specular scattering coefficient[2,12].

It is noted that Γ is a function of the RF ray grazing angle, ψ, sea dielectric constant, sea conductivity, and λ. For vertical polarisation | Γ| has a minimum at the Brewster's angle ~6-7° (for X band on a sea surface).

The coherent scattering coefficient ρ_c is based on Beard's experimental data[1,2] as a function of the sea roughness parameter *g*. The definition of *g* is:

$$g = \frac{\sigma_h \sin(\psi)}{\lambda} \quad (4)$$

where σ_h is the RMS sea wave height. Note that the expression for ρ_c has been validated for the case where $g \le 0.3$, as shown in Figure 3. Because there are no available validated ρ_c models for $g > 0.3$, we have followed Northam's approach in the present simulation by extending the upper limit to include all values of *g*.

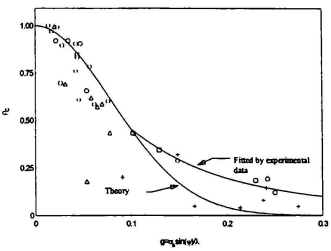


Figure 3. *Coherent field versus apparent ocean roughness*

2.4 Incoherent Component

In model 1, the incoherent component is stochastic in nature and is assumed to be the resultant sum of vectors from many, independent, random scatterers. Therefore, *I* is represented as the sum of two zero mean Gaussian processes, I_P and I_Q, which are orthogonal. The voltage of diffuse or incoherent signal component is

$$I = \{100P_{Tx}G_{TxC}G_{RxC}\lambda^2/[4\pi(R_{IS}+R_{SM})]^2\}^{1/2}\, \sigma_i\, (I_P + jI_Q)$$
$$\sigma_i = |\Gamma|\rho_i\underline{D} \quad (5)$$

where I_P and I_Q are generated by Gauss-Markov process[2,12], and ρ_i represents the diffuse scattering coefficient. Figure 4 illustrates the ρ_i model (lines) against validation experimental data (dots). It is noted again that the validation of the ρ_i model is also limited to *g* less than ~0.3.

In model 2, the incoherent component is

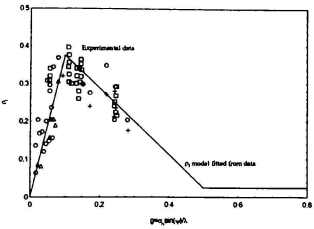


Figure 4. *Incoherent scattering coefficient.*

$$I = \int\int \frac{100P_T G_{TXI} G_{RXI} \lambda^2 \beta_0 \sqrt{[1-\rho_c(\psi_{in})][1-\rho_c(\psi_{out})]}}{(4\pi)^3 R_{ID}^2 R_{DM}^2}\, \Gamma\, \underline{D}\, exp(-j\alpha_I)\, dA + \left\{\int\int \frac{100P_T G_{TXQ} G_{RXQ} \lambda^2 \beta_0 \sqrt{[1-\rho_c(\psi_{in})][1-\rho_c(\psi_{out})]}}{(4\pi)^3 R_{ID}^2 R_{DM}^2}\, \Gamma\, \underline{D}\, dA\right\} (J_P + jJ_Q)$$

$$\alpha_I = 2\pi(R_{ID} + R_{DM} - R_{IM})/\lambda - \theta_I \qquad (6)$$

where J_P and J_Q are N(0,1) normally distributed random signals filtered by a band-pass filter centered at $\mathbf{V_S} \bullet \mathbf{e_w}/L_w + f_w$ with bandwidth f_{RS}[2,12], $\mathbf{V_S}$ is the velocity of the specular point, $\mathbf{e_w}$ the horizontal unit vector of the wind direction, and L_w the sea wavelength, f_w and f_{RS} are center frequency and frequency band-width determined by sea state[2,12], R_{ID} and R_{DM} are the ranges from a scattering patch to both Tx and Rx, θ_I the Tx phase at the off-boresight angle of R_{ID}, ψ_{in} and ψ_{out} the grazing angles formed by R_{ID} and R_{DM}, and β_0 the RMS sea surface wave slope[14]. The antenna gain patterns in (6) (G_{TXI} and G_{RXI}) are associated with individual patches rather than the specular point (e.g. G_{TXC} and G_{RXC} in equation (5)). In equation (6), the scattering coefficient for the surface patch area dA at the given grazing angle is simplified to[4]

$$\sigma^o \sim \beta_o \quad \text{for} \quad 0 \le \beta \le \beta_o, \quad \beta = |\psi_{in} - \psi_{out}|/2 \qquad (7)$$

Outside of the glistening surface, this approach may become invalid.

In equation (6), the first term of I describes the Doppler frequency spread caused by the incoherent scattering, and the second term corresponds to a frequency shift caused by the sea wave.

2.5 Direct Calculation of Scattering Signal

In models 3-5, the scattering signal, S, is directly integrated from a ground grid. Three models represent different ground grid meshes.

In models 3 and 4, the voltage of the scattering signal is

$$S = \int\int \frac{100P_T G_{TXS} G_{RXS} \lambda^2 \sigma^o}{(4\pi)^3 R_{ID}^2 R_{DM}^2}\, \underline{D}\, exp(-j\alpha_s)\, dA + \left\{\int\int \frac{100P_T G_{TXS} G_{RXS} \lambda^2 \sigma^o \sqrt{[1-\rho_c(\psi_{in})][1-\rho_c(\psi_{out})]}}{(4\pi)^3 R_{ID}^2 R_{DM}^2}\, \underline{D}\, dA\right\} \times (J_P + jJ_Q)$$

$$\alpha_s = 2\pi(R_{ID} + R_{DM} - R_{IM})/\lambda - \theta_s \qquad (8)$$

where again G_{TXS} and G_{RXS} are antenna gain patterns associated with individual patches. A more general model for σ^o is used in calculation of S. The detailed description of this σ^o model is available elsewhere[11]. However, it is important to note that the assumption for the σ^o model relies on the following condition concerning the sea wave height and the signal wavelength:

$$\sigma_h >> \lambda. \qquad (9)$$

Model 5 differs from models 3 and 4 in that the isodop contours are used to generate the ground grid. In this model, it is convenient to integrate signal power in the frequency domain. Therefore, we calculate two scattering powers as:

$$P_{S1}(f) = \int_{\text{along } f\text{-isodop}} \frac{100P_T G_{TXS} G_{RXS} \lambda^2 \sigma^o}{(4\pi)^3 R_{ID}^2 R_{DM}^2}\, \underline{D}^2\, dA$$

$$P_{S2}(f) = \int_{\text{along } f\text{-isodop}} \frac{100P_T G_{TXS} G_{RXS} \lambda^2 \sigma^o \sqrt{[1-\rho_c(\psi_{in})][1-\rho_c(\psi_{out})]}}{(4\pi)^3 R_{ID}^2 R_{DM}^2}\, \underline{D}^2\, dA \qquad (10)$$

and then transform these two powers from the frequency domain into equivalent voltages in the time domain:

$$S_1(t) = F^{-1}[(u_r + ju_i)\sqrt{P_{S1}(f)}]$$

$$S_2(t) = F^{-1}[\sqrt{P_{S2}(f)}] \qquad (11)$$

where F^{-1} denotes the inverse FFT, and u_i and u_r are uniformly distributed random numbers with $u_i = (1-u_r^2)^{1/2}$. Finally, the total voltage of the signal S is

$$S(t) = S_1(t) + S_2(t)\,(J_P + jJ_Q) \qquad (12)$$

For details on how to calculate equation (10) using the isodop grid method, please refer to papers by

Saylor[10,11] and consider the negative Doppler frequencies on the rear receiver of the missile system.

2.6 Ground Grid Comparison

In models 2-5, either *I* or *S* signal is integrated from patches on the ground grid. Figure 5 compares the grids used in these models.

The glistening surface boundary is defined by

$$y(x) = \pm \frac{x(R-x)}{R}\left\{\frac{h_t}{x} - \frac{h_M}{R-x}\right\}\sqrt{\tan^2\beta_o - \frac{1}{4}\left[\frac{h_t}{x} - \frac{h_M}{R-x}\right]^2} \quad (13)$$

where x is the horizontal range from Tx along the Tx-to-Rx direction, y the horizontal cross range relative to the x direction, R the horizontal range between Tx and Rx, and h_t and h_M the Tx and Rx heights. The extended glistening surface covers a wider area which encloses the normal glistening surface. We use equation (12) again to determine the extended glistening surface boundary but this time β_o in (12) is replaced by $2.5\beta_o$.

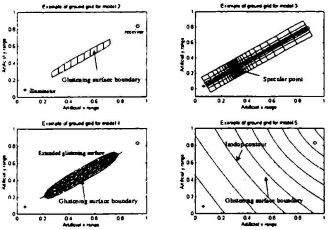


Figure 5. *Comparison of ground grid in signal integration.*

It is seen that the grid associated with models 2 and 4 focus on the glistening surface and the specular point is also included. This type of grid provides a better estimation of the diffuse signal component. The grid of model 3 has a log scale in both the x and y directions around the specular point, and aims to provide a good estimation of both the specular and diffuse components.

The isodop grid depends on iso-Doppler contours and extends over a much wider area. This implies that this grid is not efficient in the calculation of scattering power, and major signal contributions from the glistening surface surrounding the specular point are not weighted enough in the grid. Therefore this grid may lead to large errors in the integration.

3. COMPARISON OF SIMULATION RESULTS

Previous efforts[12] focused on the comparison of two multipath models from a theoretical point of view. In this paper, more multipath models are compared and attention is directed to application results. The five multipath models have been implemented using MATLAB on a Pentium PC and an evaluation of the generated signals has been performed with consideration of the following aspects:

- Model limitation.
- Model accuracy.
- Model capability to represent dynamic change of the multipath signal.
- Cross model calibration.
- Computation speed.

It is noted that we did not compare the simulated signal with experimental data. Therefore, apart from cross-validation of models, this comparison study does not contain model validation.

3.1 Comparison Baseline

Table 1 summarises the parameters used for comparison computation.

Table 1: *Parameters used for computation*

Wave length	0.03m (X-band)
Tx Power	2000W
Tx and Rx gain pattern	sinx/x with main lobe beamwidth of 2° and 75° respectively
Sea dielectric constants	80-j240λ (ε_r=55-j30, μ_r=1)
Geometry update rate	1 msec
Number of patches in glistening surface (model 2)	10
Dimension of ground grid (models 3-4)	64x32
Number of frequency bins (model 5)	64
Number of range bins along isodop (model 5)	32

The baseline missile trajectory and missile velocity profile are obtained from the 6DOF modelling of a generic RF guided missile. Figure 6 plots the trajectory. The total flight time is about 9 sec. The comparison is based on an open loop computation of the multipath signals in accordance with this trajectory and velocity profile.

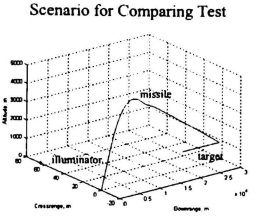


Figure 6. *Trajectory used for comparison.*

3.2 Comparison of Multipath Time Series

The final signal received at Rx is T. This could be a set of time series (voltage amplitude and phase) in the time domain, or a spectrum in the frequency domain.

For an easy comparison, the time series T is used. The amplitude of T is calculated as

$$Power = 20log_{10}(|T|), \quad in\ dB \tag{14}$$

where T is in voltage, and phase as

$$Phase = tan^{-1}(Im(T)/Re(T)) \tag{15}$$

Figures 7-9 shows the total signal amplitude and phase in the presence of multipath. Three different sea states are considered in these comparisons. When sea state equals one, a relatively smooth sea surface is represented, while sea states three and five represent medium and high sea roughness[2,12], respectively.

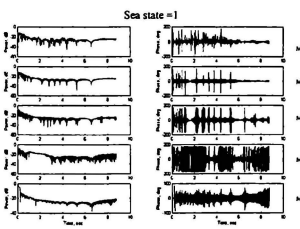


Figure 7. *Amplitude and phase of multipath signals calculated from five models. The sea state is one.*

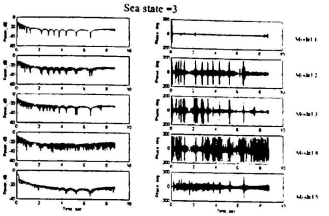


Figure 8. *Same as Figure 7, but sea state=3.*

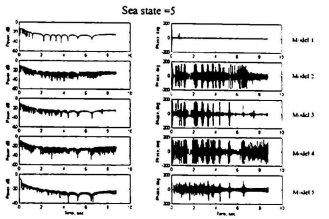


Figure 9. *Same as Figure 8, but sea state=5.*

From comparing the time series results of the five models, the following features are apparent:

- Models 2 and 3 generate similar multipath signal in terms of both amplitude and phase. Note that Model 2 separates C and I in S while model 3 does not. This implies that if the ground grid is set up properly, different approaches can give a very close result. However, a larger difference is seen when the sea state becomes larger. This may be due to ρ_C, which is based on the fitting of experimental data, failing to encapsulating the more realistic situation. This will be further discussed later.
- Similar to models 2 and 3, model 1 gives reasonable output for low sea state, but for high sea state, the diffuse component obtained from model 1 is significantly under-estimated. The C component may also be under-estimated. Again, the reason for this may be that for the high sea states ρ_I and ρ_C, which are based on the fitting of experimental data, may not valid. Again, this will be discussed later.
- Model 4 exhibits features similar to model 3. However, when sea state is low, a stronger incoherent component with large phase fluctuations is seen. This is not surprising as the ground grid for this model has been designed to focus more on the diffuse signal.

- Model 5 demonstrates significantly different features from the other four models in both amplitude and phase. This may be due to the large calculation errors caused by a poor design of the ground grid.

It is interesting to note that the different models have varying strong and weak points when estimating the specular and diffuse components of the scattering signal. In model 1 both components are limited to small g (~0.3), and this implies low sea state in the considered scenario. An extension of the ρ_i and ρ_e models is needed to cover a general missile engagement scenario. In model 2, the specular component has the same problem as that in model 1. But the diffuse component is applicable to more general cases. Model 3 has a fine grid mesh surrounding the specular point thus ensuring a more accurate specular component estimation. In addition, model 3 includes an extended mesh to the nearby area for a fair estimation of the diffuse component. Model 4 is similar to model 3, but the grid mesh is not particularly focussed on the specular point and therefore is more general for the diffuse component. Model 5 attempts to use frequency bins to collect the scattering power, but the grid mesh is very crude and consequently under represents the major contributing area to the power. Hence, this model causes large computational errors.

3.3 Comparison of Multipath Spectra

There are two types of spectra used in the comparison. One is a static spectrum which is generated via the FFT of a nine sec strip of each model's time series data output:

$$\textit{Static spectrum} = 20\log_{10}(|F(T)|), \textit{ in dB} \qquad (16)$$

where F represents the FFT.

Another is a dynamic spectrum with a moving FFT window. In the dynamic spectrum, the spectral power in dB is plotted as the grey scale: the darker the colour the higher the power. The moving window length is 0.5 sec, and the moving window overlap is 90%.

The static spectrum is useful for examining the power level in the signal while the dynamic spectrum provides an indication of overall model response to dynamic changes in geometry and other scenario parameters. Figures 10-12 summarise the signal spectra from each of the five models. The following features are seen in the comparison:

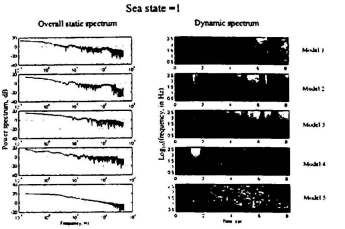


Figure 10. *Static and dynamic spectra of the total signal. The sea state is one.*

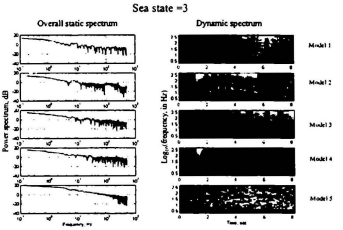


Figure 11. *Same as Figure 10, but sea state =3.*

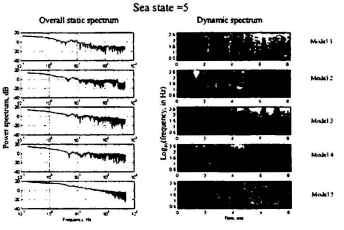


Figure 12. *Same as Figure 11, but sea state =5.*

- In the static spectrum the specular component is shown as a low frequency peak at 5-10 Hz. This peak does not appear clearly in the dynamic spectrum because its frequency is too low for the moving FFT window. In model 5 this peak disappears too, because of such a poor ground grid mesh that the specular component is not properly calculated.

- In all sea states, models 1-4 exhibit a similar level of power, while in model 5 the signal spectral power level is generally a few dB higher. This may also be due to the larger scattering power estimation error in model 5.
- At high frequencies, the static spectrum of model 1 differs from those of models 2-4. This corresponds to different methodologies in the scattering signal computation.
- At low frequencies, the static spectral feature is mainly determined by the direct signal specular component of the scattering signal. In models 1-4 these features are very similar.
- Model 2-4 show good and similar dynamic responses in their spectra, while model 1 only shows significant dynamic response when the sea state is low. Model 5 is poor in dynamic response.

4. DISCUSSION

In this section further details of the model applications will be discussed.

4.1 Model Validation Check

As we mentioned before, models 1 and 2 are based on the experimental data fitting of ρ_c and ρ_i validated up to g~0.3. It is interesting to note that for a general missile trajectory such as the one used for the comparisons, a small g implies a low sea state. Figure 13 shows g, ρ_c and ρ_i variation over the whole trajectory when sea state equals one and five. For the missile example demonstrated here, the flight height reached its maximum at ~ 1.7 second when g also reached its maximum. For sea state one, the maximum g is ~0.35 and therefore ρ_c and ρ_i calculated from models 1 and 2 appears in the experimental data ranges (compare Figures 3 and 4). However, when the sea state equals five, g can climb as high as 18, and in most trajectories g >>0.3, well above the validated data range. In this case, ρ_c~0 and ρ_i=0.025 in our model. This explains the results in figure 9 (sea state=5) whereby model 1 exhibits less multipath variation than for a lower sea state situation.

Figure 14 summarises the validated g range and the g practical range reached during our simulation example. It is seen that in order to increase our confidence using models 1 and 2, more data collection trials would be required to widen the validation area of figure 14.

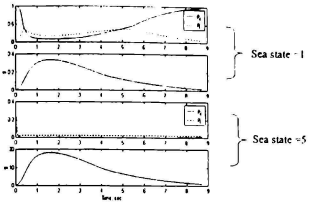


Figure 13. *Scattering coefficients in models 1 and 2.*

In general, a large g normally implies a large grazing angle since the sea state refers to a fixed ratio σ_h/λ. In this paper we focussed on multipath effects as seen at the missile rear receiver. For the missile front receiver, the effects of multipath are also important. Let us consider a special case of the front receiver multipath: the target scatters the illuminating wave to the sea surface and then the sea surface scatters it forward to the front receiver. In this case the multipath appears as an "image" of the target to the front Rx with the sea surface as a mirror. For a missile front receiver, the multipath signal will be a superposition on the target signal, and may cause significant angle tracking fluctuations. For low altitude targets, the grazing angle for the front Rx multipath is generally small over the missile trajectory. However, during the final stage of terminal phase, the grazing angle may become substantially large. In this case, the range and directional discrimination between the target and multipath image is relatively large too. This may enable the seeker signal processing to separate and cancel the multipath signal. This example may reduce the importance of the large grazing angle for the front receiver, but for the rear receiver the effect of a large grazing angle is still critical.

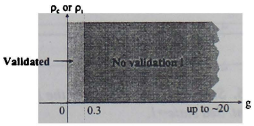


Figure 14. *Models 1 and 2 validation check.*

4.2 Statistical Distribution Calibration

One consideration when developing a complex, accurate but relatively slow multipath model is to calibrate a fast, simpler model. The calibrated simpler model can then serve as the basis for real-time digital or hardware-in-the-loop simulation. Model 1 could serve as the candidate for calibration as it is the more efficient and least computer intensive of the five models.

There are two parts to be calibrated in model 1: C and I. Calibration of C is the same as validation of ρ_c, and was discussed in the above subsection.

In model 1, the I compoment is generated by a Gauss-Markov process which filters Gaussian distribution (N(0,1)) noise with center frequency at f_c and bandwidth f_{RS} [12]. Now, in order to calibrate this model the first thing is to check the random number generator: is it Gaussian or other distribution? Figures 15-17 are plots of the amplitude distribution of I or equivalent I component in each of the five models. The equivalent I component in models 3-5 is calculated by the following method. If the scattering signal, S, is denoted as

$$S = S_I + S_2 (J_P + j J_Q)$$

Then the equivalent I is

$$I \sim S_2(J_P + jJ_Q) + \int [1 - \rho_c(\psi_{in})][1 - \rho_c(\psi_{out})]^{1/4} s_1 dA \quad (17)$$

where s_1 is the voltage contribution to S_1 from the surface patch dA.

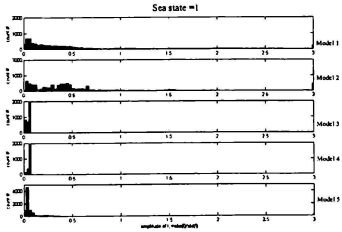


Figure 15. *Statistical distribution of the I component. Sea state=1.*

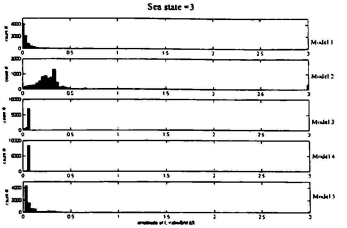


Figure 16. *Same as Figure 15, but sea state=3.*

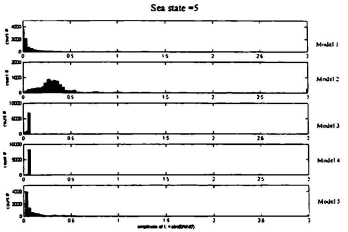


Figure 17. *Same as Figure 16, but sea state=5.*

It is seen from figures 15-17 that for all sea states, the diffuse component amplitude distribution differs. As expected, model 1 is associated with a Gaussian-like distribution. However, the other model distributions can hardly be described as Gaussian. Unfortunately, this makes distribution calibration difficult.

4.3 Choice of the Model

Choosing a model for RF multipath signal generation in missile performance studies depends on many factors. Here we have studied the effect of multipath on the rear receiver signal of a semi-active missile for a given flight scenario. From our studies, it is clear that different models have different advantages and disadvantages. Figure 18 attempts to summarise the results based on this comparative study.

It seems that model 5 has performed poorly in comparison because of the rough design of the ground grid. The other four models represent the basic features of the multipath signal, but still have their practical limitations.

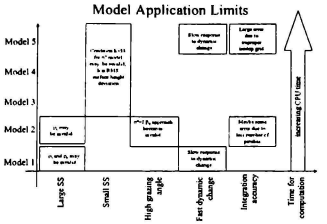


Figure 18. *Some concerns in choosing a model.*

Model 1 has a problem in the large g regime. This limits its application for rear receiver. But for the front receiver with low level targets, it could perform better.

Model 2 improves the diffuse component generation. But the coherent component may still be limited at low grazing angles.

Model 3 covers both coherent and diffuse components well by the special design of the ground grid. Its scattering coefficient model requires a rough sea surface. This model runs slower than models 1 and 2.

Model 4 is similar to model 3. However, the ground grid design differs, so that it emphasises more of the diffuse component. The representation of the coherent component may be less accurate.

It is noted that atmospheric ducting effect is not included in these models. In fact, ducting plays an important role in RF wave propagation. Comparison of a parabolic wave equation model will be the topic of a future paper.

5. CONCLUSION

In this paper, we have presented a brief description of several multipath models that can be used in conjunction with a missile 6DOF simulation model for assessing the performance of radar guided missiles against sea skimming threats. Both qualitative and quantitative comparisons of the various models have been made. The numerical results are based on a typical scenario concerning the protection of a Naval ship against an attacking low level threats. Model choice, limitation, and other application concerns have been discussed through numerical result comparison.

6. REFERENCES

1. C.I. Beard, I. Katz & L.M. Spetner, "Phenomenological Vector Model of Microwave Reflection from the Ocean," IRE Trans., **AP-4**, pp. 162-167, Apr 1956.
2. D.Y. Northam, "A Stochastic Simulation of Low Grazing Angle, Forward Scatter, Over-Water Multipath Effects," NRL Report 8568, Dec 1981.
3. P. Beckmann & A. Spizzichino, "The Scattering of Electromagnetic Waves from Rough Surfaces," Pergamon Press, 1963.
4. D.K. Barton, "Low Angle Tracking," Proceedings IEEE, **Vol. 62**, No 6, pp. 687-704, June 1974.
5. R.M. Smith, J.Y. Yee, C.S. An & A.L. Haun, "Simulation of Multipath for Semi-Active Missiles," AGARD CP-473, 1990.
6. J.L. Lopez, "Multipath Model," TSI Technical Memo TD-85-13, 1985.
7. J.H. Osborne, "NSWCDD Proposed Multipath Model for the Close-In Weapon System (CIWS)," NSWCDD-TR-92-35, 1992.
8. J. Chisholm, "Incorporation of a Frequency-Domain Diffuse Multipath Model into a Comprehensive Multipath Simulation," Unpublished Report, NSWC Dahlgren, 1996.
9. J.S. Silverstein, J. Saffold & M. Christian, "Evaluation of the TKMOD5C Multipath Simulation Code," Conference Paper at GTMV, August 1996.
10. A.V. Saylor, "A Real-Time CW Clutter Model for Semi-Active Systems," Simulation Technologies Inc., Tech Note 130-008, Aug 1984.
11. A.V.Saylor, "Validation of Analytical Forward-Scattering Prediction Model," SCSC Proceedings, 1992.
12. D. Bucco & J. Chisholm, "Comparison of Scattering Models for Predicting Radar Multipath Effects over the Sea," AIAA MST Conf., New Orleans, USA, Aug 1997.
13. M.W. Long, "Radar Reflection of Land and Sea," Lexington Books, Lexington, Mass., 1975.
14. R.L. Munjal, "Comparison of various multipath scattering model," JHU/APL Memo F1D-0-86U-111, 1986.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37249

AIAA-2000-4288

MISSILE AERODYNAMIC IDENTIFICATION USING MIXED EKF AND EBM

Ali Mohammadi*
YMA college, PO. BOX. 19585-774, Tehran, IRAN
and
Mohammad Ali Massoumnia**
Sharif University of Technology, Tehran, IRAN

Abstract

In this paper, we review recently developed classification schemes used in Aerodynamic Identification based on flight test data (EBM and EAM) and present a new classification from identification point of view (Identification With/After Smoothing (IW/AS) and Sequentially Smoothing and Optimization (SSO)). Also, we present a new method to account for unmodeled forces and moments in the system dynamics and assess its performance using several simulations. Using this method, EKF Smoothing based on Information Square Root Filtering (ISRF) is applied to an Antitank Guided Missile (AGM) aerodynamic identification problem. As we know, the efficiency of the applied algorithms is very critical in this application, since the missile velocity changes from mach 0.2 to 0.92, aerodynamic parameters can not be assumed constant during the flight. Simulations show promising results in different flight conditions.

I. Introduction

Aerodynamic model and its accuracy have critical role in guidance and control design of aircraft, helicopter and missile. Parameters of this model are usually obtained by numerical methods or from wind tunnel test data. In the past decades, system identification methods are applied in many engineering fields such as aerodynamic parameter identification and have shown their valuable potentials. For example, Norton[6,7] in 1920's extracted the natural frequency and damping ratio of an aircraft using flight test data. Some of the reasons for using flight test data in aerodynamic identification are as follows:

1. Wind tunnel tests are carried out under restricted conditions and simulating all possible flight conditions is quite difficult.
2. Aerodynamic parameters that are estimated from flight test data can be used in extending the flight regime of the flying vehicle.
3. These techniques can be used for reliable evaluation of the parameters obtained from wind tunnel test.
4. A different application of aerodynamic parameter estimation is in tracking filters. For example in a defense system which will be used against targets with unknown or poorly known aerodynamic parameters[8].

In this paper different methods in aerodynamic parameter identification are assessed. In section II, based on an extensive search, a recent classification of different identification methods is presented. After comparing different schemes, we present a new classification from identification point of view. In section III Extended Kalman Filter (EKF) is chosen as a smoothing algorithm for parameter estimation. Since aerodynamic models are usually not very accurate, and aerodynamic forces and moments are not well characterized in our model, we should some how account for modeling errors in our estimation scheme. In section IV, we present a new strategy to account for unmodeled dynamics. In section IV we apply our EKF-smoothing based strategy to a Medium Range AntiTank Missile (MRATM). Simulation results are explained in details and different real field scenarios are considered.

II. Aerodynamic Identification Methods

Formulation

It is well known that the six degrees of freedom dynamics of a missile is as follows:

---

* Ph.D. student, Dept. of Electrical Engineering, Sharif University of Technology
** Associate Professor, Dept. of Electrical Engineering, Sharif University of Technology

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

$$\dot{\underline{v}}^B = (\underline{F}_a^B + \underline{F}_t^B)/M + C_I^B \cdot \underline{g}^I - \underline{\omega}^B \times \underline{v}^B$$

$$\dot{\underline{r}}^I = C_B^I * \underline{v}^B$$

$$\dot{\underline{\omega}}^B = I_N^{-1}\left[\underline{M}_a^B + \underline{M}_t^B - \underline{\omega}^B \times (I_N * \underline{\omega}^B)\right]$$

$$\dot{\underline{\gamma}} = \begin{bmatrix} 1 & Sin\phi * \tan\theta & Cos\phi * \tan\theta \\ 0 & Cos\phi & -Sin\phi \\ 0 & Sin\phi/\cos\theta & Cos\phi/Cos\theta \end{bmatrix} * \underline{\omega}^B \qquad (1)$$

There are always in a missile or an aircraft, instruments that measure (in general functions of) the above state variables. These measurements can be modeled as follows:

$$\underline{a}_m^B = (\underline{F}_a^B + \underline{F}_t^B)/M + \underline{\omega} \times (\underline{\omega} \times \underline{r}_S) + \dot{\underline{\omega}} \times \underline{r}_S + \underline{n}_a$$

$$\underline{r}_m^I = \underline{r}^I + \underline{n}_r$$

$$\underline{\gamma}_m = \underline{\gamma} + \underline{n}_\gamma \qquad (2)$$

$$\underline{\omega}_m^B = \underline{\omega}^B + \underline{n}_\omega$$

Note that, in missile applications, depending on the mission, measuring instruments are usually cheaper or simpler than those used in aircraft applications.

Next, we model aerodynamic force and moment vectors as follows:

$$\underline{F}_a^B = QA\underline{C}_F^B = 1/2 * \rho A \underline{v}^2 . \underline{C}_F^B$$

$$\underline{M}_a^B = QAD\underline{C}_M^B = 1/2 * \rho AD \underline{v}^2 . \underline{C}_M^B$$

$$\underline{C}_F^B = \begin{bmatrix} C_X \\ C_Y \\ C_Z \end{bmatrix} \quad \underline{C}_M^B = \begin{bmatrix} C_L \\ C_M \\ C_N \end{bmatrix} \qquad (3)$$

Assuming a constant flight altitude, the density ρ can be assumed constant. In general, the components of $\underline{C}_F^B$ and $\underline{C}_M^B$ are nonlinear functions of state variables and their derivatives. But in many cases, aerodynamic model can be written as follows:

$$C_X = C_{D0} + C_{a_C} \cdot (\alpha^2 + \beta^2)$$
$$C_Y = C_{Y\beta} \cdot \beta + C_{Yr} \cdot r + C_{Y\delta_r} \cdot \delta_r$$
$$C_Z = C_{Z\alpha} \cdot \alpha + C_{Zq} \cdot q + C_{Z\delta_e} \cdot \delta_e$$
$$C_L = C_{Lp} \cdot p + C_{L\delta_a} \cdot \delta_a \qquad (4)$$
$$C_M = C_{M\beta} \cdot \beta + C_{Mr} \cdot r + C_{M\delta_r} \cdot \delta_r$$
$$C_N = C_{N\alpha} \cdot \alpha + C_{Nq} \cdot q + C_{N\delta_e} \cdot \delta_e$$

Where C_{ij} here are functions of object velocity. In an antitank missile with a velocity variation in the range of mach 0.2 to 0.92, these parameters can not be assumed constant. Hence, our problem has the general form of a nonlinear estimation problem modeled as follows:

$$\dot{\underline{X}} = f(\underline{X}, \underline{\theta}(t), \underline{u}(t), \underline{w}(t), t) \qquad (5)$$

$$\underline{Y}(t_i) = h(\underline{X}(t_i), \underline{\theta}(t_i), \underline{u}(t_i), \underline{w}(t_i), t_i) \qquad (6)$$

Where $\underline{x}$ and $\underline{\theta}$ are state variables and aerodynamic parameters.

EAM and EBM

In this section we elaborate on the differences between Estimation After Modeling (EAM) and Estimation Before Modeling (EBM) approaches to parameter estimation. EAM is based on modeling the aerodynamic forces and moments and estimating the parameters of this model alongside the missile dynamics. EBM is based on estimating the missile dynamics, aerodynamic forces and moments and then fitting an appropriate model to these estimated forces and moments. Note that the stage in which the aerodynamic forces and moments are modeled during the estimation procedure, distinguishes these two techniques from each other. The block diagram of these two techniques are shown in Fig. 1 and Fig. 2.

As is clear, EBM is an open loop estimation procedure. So for example, error in estimating angle of attack does not affect the error in estimating aerodynamic forces and moments. However, EAM does not have this shortcoming and simulation results also confirm the superior performance of EAM compared to EBM.

Also note that in EBM, aerodynamic forces and moments are considered as Markov processes during the estimation process. It can be shown that process noise in this method is correlated with states[20], and hence, the uncorrelatedness assumption between states, process noise and measurement noise used in almost all identification methods is violated[⊕].

Identification Methods

In the literature, many methods are used to estimate aerodynamic parameters using flight test data. Some of these methods are as follows: Output Error[10], Equation Error[5,10], Equation Decoupling[18], Nonlinear Smoothing Identification (like Extended Kalman Filtering)[9,11,13,17,21,22], Maximum Likelihood[5,12,14,15,16,19] and Smoothing based

[⊕] Idan[20] used another formulation to avoid this problem. In that formulation, it is assumed that angular velocity vector and linear acceleration of C.G. are measured. It can be shown that if we can not measure the complete these vectors, Idan's formulation can not be used.

Identification[20,23]. The later three methods are used extensively.

In Smoothing Identification methods, since parameter vector θ is considered constant, the procedure is divided to two stages. In the first stage, states are estimated, and in the second, parameters are updated such that a cost function J is minimized. Estimation (or smoothing) of states is performed via Maximum A Posteriori[22] (MAP) smoother or EKF (like Modified Bryson-Frazier (MBF)), with a backward-forward filtering. Update of the parameters in the second stage is performed with an optimization algorithm. Finding the gradient of the cost function J with respect to parameters is a critical task here. In ML, this task is performed by computing the sensitivity matrix[14]. But in smoothing based identification methods, the gradient is computed efficiently using Langrange multipliers during smoothing procedure. Whenever the parameters are updated, the smoothing algorithm is performed once again. This procedure is repeated until changes in parameters are very small.

In a general case, ML estimation can be considered as a member of smoothing based identification class, if process noise is also included.

We name this class of algorithms as sequentially smoothing and optimization (SSO). It is clear that global optimization with respect to states and parameters, is broken to two successive optimizations.

In another identification class, parameters are modeled as Markov processes and are augmented to state variable vector. Hence the problem of parameter identification becomes a nonlinear smoothing (or filtering in on-line applications) problem. These methods are generally extended from linear state estimation theory.

We call this class as Identification With/After Smoothing (IW/AS). Time varying parameters are more easily identified in this class. Of course like EBM, we can model external forces and moments in nonlinear state estimation stage and estimate them. In an auxiliary step, a regression method is applied to find unknown parameters.

The degree of nonlinearity and dimension of matrices that we countered in IW/AS class are higher than those used in the first stage in SSO class. SSO methods include an iterative smoothing and optimization algorithms until final results are obtained. Various smoothing and optimization algorithms can be used in two stages of SSO methods. But in IW/AS class, an algorithm is performed only once on the given data.

III. Frazier Smoother based on Square Root EKF

We have selected Extended Kalman Filtering as parameter estimator from IW/AS class. Major Kalman filter (KF) smoothing algorithms are MBF[28] and Rauch,Tung, and Stribel (RTS)[28]. Here, we use Square root (SR) algorithm for implementation of KF. Numerical difficulties that appear in other implementations are not seen in SRKF[29]. Square root algorithms can be divided to two different categories: Covariance SRKF and Information SRKF. Since covariance of backward filtering at $t=t_N$ must be $P(t_N)=\infty$[30], ISRKF algorithm is superior in this regard, since $\Lambda(t_N)=P^{-1}(t_N)=0$. Frazier smoothing algorithm based on ISRKF is shown in Fig. 3.

The following notes should be considered:

1) In a theoretical view, it can be shown that Kalman Filtering as a parameter estimator fails with a probability of one. But in a real world application this is case dependent. In many papers, EKF is used to identify aerodynamic parameters using flight test data. In a comparison[11] with Nonlinear Recursive Prediction Error Method and ML, it is shown that EKF is superior.
2) A disadvantage of this method is the increase in the number of states after augmenting the unknown parameters to the state vector.
3) The conventional estimate of R (if measurement noise is white Guassian) using the N most recent residuals, is given by

$$\hat{R} = \frac{1}{N-1}\sum_{i=1}^{N}(\underline{\nu}_i - \bar{\nu})(\underline{\nu}_i - \bar{\nu})^T \qquad (7)$$

where

$$\bar{\nu} = \frac{1}{N}\sum_{i=1}^{N}\underline{\nu}_i \qquad (8)$$

Robust versions of eq. (7) (with respect to outliers and the assumption that $\underline{\nu}_i$ are white Guassian noise) can be found in [31,32].

4) Optimal estimation of Q is a difficult task. It has been shown that under special conditions, it is possible to optimally estimate Q adaptively[1]. This technique is named as Residual Correlation Method (RCM). The most important restrictions in this method are:
   a) System must be LTI.
   b) In order to find a unique solution, the number of unknown components in Q must be less than n*m (number of states multiplied by number of inputs).

We can not use RCM because our model is not LTI. But tuning of Q based on whiteness of residuals is the essential idea that is used in all Q estimation methods. The relationship between Q and Residuals can not be derived analytically, but several different empirical relationships are proposed[2]. A small Q derives residuals unwhite. A large Q make residuals white but parameter estimate degrade from their real values. Hence small Q must be used at first. Then

gradually increase Q such that all residuals are whitened. However, changes made to individual elements of Q are based on engineering experience. For example Q_{22} (process noise component in dv/dt equation) affects the a_1 accelerometer residuals. And Q_{11} (process noise component in dw/dt equation) affects the a_2 accelerometer residuals. These facts help us in tuning Q components. Otherwise, handling of Q components separately is usually a very difficult task.

IV. Unmodeling effects

It is clear that aerodynamic parameter estimation is based on system dynamic and aerodynamic models. In real world, these models may be inaccurate. In dynamic models, assumption such as:

1) Product moments of inertia are assumed zero, since the missile is assumed cruciform.

2) Thrust curve is known.

3) (u,v,w) are linear velocity in body axis coordinate. If wind is absent, angle of attack and sideslip will be

$$\alpha = \tan^{-1}(\frac{w}{u}), \beta = \tan^{-1}(\frac{v}{u}) \quad (9)$$

If wind flows, angle of attack and sideslip become

$$\alpha' = \tan^{-1}(\frac{w - w_w}{u - u_w}) \quad (10)$$

$$\beta' = \tan^{-1}(\frac{v - v_w}{u - u_w})$$

since

$$u_w, v_w, w_w, v, w << u$$

above equations can be approximated as

$$\alpha' \approx \frac{w - w_w}{u - u_w} = \alpha - \frac{w_w}{u}$$

$$\beta' \approx \frac{v - v_w}{u - u_w} = \beta - \frac{v_w}{u}$$

so we have

$$C_z = C_{z\alpha}\alpha + C_{zq}q + C_{z\delta_e}\delta_e \quad \text{assumed model}$$

$$C_z' = C_{z\alpha}\alpha' + C_{zq}q + C_{z\delta_e}\delta_e \quad \text{real model}$$

Hence an unmodeled dynamics is produced.

4) Every manufactured subsystem of a missile has its tolerances. For example misalignment of motor nozzles can produce a new force and moment around C.G. of the missile. These forces and moments usually differ from one missile to another in actual flight.

5) A linear aerodynamic model or a deterministic nonlinear aerodynamic model is usually assumed. If this model is not accurate, estimated parameters deviate from true values.

A Theoretical Discussion

In KF literature, it is always assumed that unmodeled dynamics can be modeled as process noise. This noise is White and Guassian (WG) with covariance matrix Q.

1) If process noise model were not WG, which is certainly the case in actual practice, KF becomes suboptimal even for linear systems.

2) If the process noise is not assumed white, but is assumed Guassian, a broader range of unmodeled dynamics can be accounted for.

3) There are filters that are less sensitive to noise statistics and unmodeled dynamics compared with KF. Generally, these filters are called Robust Estimators and can be classified as follows:

3.a)[27,26] Process and measurement noises are arbitrary signals with limited energy (based on H2 norm or H∞ norm). These filters guarantee that if energy of these signals were less than γ, their estimation error (H2 norm or H∞ norm) will be less than δ (that δ relates to γ). It can be shown that in smoothing applications, H2 and H∞ norms have identical performance.

3.b)[25] Process and measurement noises are WG, but there are uncertainties in system dynamics which can be modeled as follows:

$$\dot{X} = (A + \Delta A)X + B.U$$

$$Y = (C + \Delta C)X$$

Here, if the norm of ΔA and ΔC were less than γ, estimation error (H2 norm or H∞ norm) will be less than δ.

4) To increase the robustness of Kalman Filters, Proportional Integral (PI)[3] or Proportional Fading-Integral (PFI) Kalman filtering[4] can be used. In this method new states that describe any unknown Gaussian input, are augmented to state vector. Indeed this method equals to Disturbance Estimation.

In our application, six new states are added. These states are used to include any unmodeled external forces and moments.

Since it is proved that EAM methods are more accurate than EBM methods, it is better to use a preliminary aerodynamic model. This model is based on theoretical approach. Any other unknown forces and moments are modeled as Markov processes.

In estimation phase, external forces and moments fit themselves to a preliminary modeled section and a model-free section (Markov processes). Since this procedure, unlike EBM, is not completely open loop and uses an approximate aerodynamic model, it inherits many advantages of EAM. On the other hand, model-free section is like EBM.

After that estimation procedure is completed, if a relation between estimated unmodeled forces, moments, state variables and inputs were found, a complementary model will be obtained and will be augmented to preliminary aerodynamic model section. Repeating the above procedure with the improved model will result in more accurate

estimated parameters, due to improved performance of EAM compared to EBM.

Tuning of process noise covariance of these unknown forces and moments (Q_b) is critical. If covariance matrix Q_b is too large, the model-free section will be overestimated and estimated parameters deviate from true values. On the other hand, if covariance matrix Q_b is too small, the model-free section will be underestimated.

Hence, at first, tune Q elements to obtain maximum whiteness of residuals. If residuals remain unwhite, increase Q_b, until white residuals are obtained.

V. Simulation

We used SRKF for aerodynamic parameter estimation of a Medium Range AntiTank Missile. This MRATM has Automatic Command to Line Of Sight (LOS) guidance and is tracked with an IR tracker at the launch sight. Tracker measures Y and Z deviation of missile with respect to LOS in inertial Coordinate. Control commands are sent to missile by wires. Pulse Width Modulated actuator is used for applying aerodynamic forces and moments to force the missile on the LOS. A two-degree of freedom gyroscope measures roll and yaw angles. Two additional 3-axis accelerometers are mounted at fore and aft of the missile. This helps to measure a function of angular accelerations. Although pitch angle and angular rates are not measured, but our results show that pitch angle and other state variables can be estimated accurately. An on board high shock resistant data acquisition system is designed to record all events and outputs. After flight, the saved data are downloaded to a computer. This method is very cost effective compared to telemetry, and also there will be no interference between radio transmission and missile internal electronics. Moreover, we record range-independent high quality signals and there is essentially no limit on the number of channels that can be recorded.

Flight motor of the missile burns for about 2 sec and after that, the missile coasts. During the first phase, nozzle output jet flows through the wings. By assuming time-varying coefficients, we can track the aerodynamic parameter changes. Therefore, if these parameters differ at the same velocity (during motor burning and coasting phase), this difference can be detected. In SRKF implementation, f and h derivatives are computed by MATAB symbolic toolbox. Sampling times of instruments are not equal. Fig.4 shows comparison of EBM and EAM algorithm in estimating pitch angle, v and w linear velocities. Based on other estimated variables, EAM is superior to EBM.

In the aerodynamic model we add a disturbance term with the following model

$$C_z = C_{z\alpha}\cdot\alpha + C_{zq}\cdot q + C_{z\delta_e}\cdot\delta_e + 0.1$$

in Y and Z channel. This disturbance equals 20% of Cy or Cz. Fig. 5 compare SRKF-DE (Disturbance Estimator) and SRKF in aerodynamic parameter estimation with this disturbance. These Figures show superiority of SRKF-DE in parameter estimation. Following notes are important in implementation of RKF-DE in our application:

1) Since missile is a cruciform, following equalities are valid:

$$C_{z\alpha}=C_{y\beta},\ C_{z\delta e}=C_{y\delta r},\ C_{m\alpha}=C_{n\beta},\ C_{m\delta e}=C_{n\delta r},\ C_{mq}=C_{nr}$$

2) Estimation accuracy of $C_{z\delta}$ and $C_{m\delta}$ are much higher than other parameters. This is because of persistent excitation of control derivatives by PWM control inputs.

3) Since angle of attack of this missile is not very high, angle of attack and side-slip derivatives $C_{z\alpha}$, $C_{m\alpha}$ and $C_{d\alpha c^2}$ have inferior estimation accuracy.

4) Roll angle is stabilized with phase adjustment between pulse commands. In the initial flight phase, magnitude of input command of roll angle is high, but after a few seconds, it decreases appreciably. Therefore the roll channel is not sufficiently excited and identification becomes inaccurate. For increasing the estimation accuracy of roll aerodynamic coefficients, an exciting scenario must be designed. Please note that $C_{l\delta}$ is estimated more better than Clp.

5) Residuals are saved in forward and backward filtering. Backward residuals are more white than forward residuals. Test of whiteness is performed based on backward residuals.

6) Estimation algorithm is not sensitive to initial conditions and does not diverge in any simulated conditions.

7) An important feature of any estimation algorithm is that its ability in predicting a bound for the estimation error. In KF, square root of diagonal elements of covariance matrix is a good criteria. Fig.5 shows the 1σ Estimated Error Band (EEB) and real error between estimated parameters and true parameters in the simulation. In all estimated parameters, except $C_{z\alpha}$, $C_{m\alpha}$ and $C_{d\alpha c^2}$, EEB is greater than real error.

VI. Conclusion

In a comparison of EBM and EAM methods in aerodynamic identification, it is shown that EAM methods are superior, because of their closed loop property.

EKF method as a member of IW/AS class is chosen for time varying aerodynamic identification. This technique is augmented with a new method for aerodynamic identification of unmodeled dynamics and the performance of this technique is illustrated through simulation results.

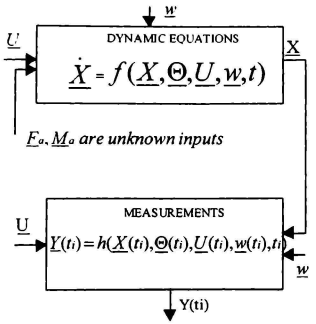


Fig. 1 Estimation Before Modeling

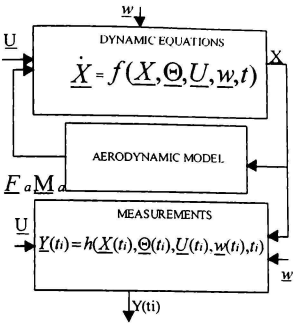


Fig. 2 Estimation After Modeling

$H = \partial h / \partial X_a$; $F = \partial f / \partial X_a$; $Q = G * G^T$; $b^- = S_{x^-} \hat{X}^-$

TimeUpdate:

$$\begin{bmatrix} N \\ b^-(k+1) \end{bmatrix} = QR\left(\begin{bmatrix} 0 \\ b^+(k) \end{bmatrix} \right)$$

$$\begin{bmatrix} N & M \\ 0 & S_{x^-} \end{bmatrix} = QR\left(\begin{bmatrix} R^{-1/2} & 0 \\ S_{x^-} F^{-1} G & S_{x^-} F^{-1} \end{bmatrix} \right)$$

Measurement Update:

$$\begin{bmatrix} S_x \\ 0 \end{bmatrix} = QR\left(\begin{bmatrix} S_{x^-} \\ G^{-1/2} H \end{bmatrix} \right)$$

$$\begin{bmatrix} b^+(k) \\ N \end{bmatrix} = QR\left(\begin{bmatrix} b^-(k) \\ R^{-1/2} Y_m \end{bmatrix} \right)$$

ISRF:

$$\hat{X}_+ = (S_f S_f^T + S_x S_x^T)^{-1} (S_f \hat{b}_f + S_x \hat{b}_x)$$

where

QR: QR decomposition, ;

S_f and S_x are square root inverse of Covariance matrix

Fig. 3 Square Root Extended Kalman Smoothing

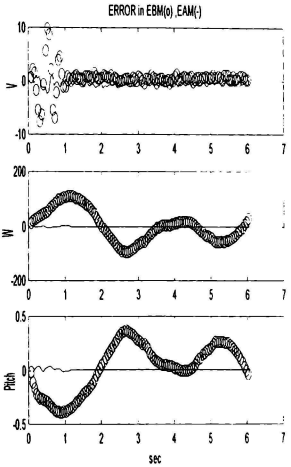


Fig.4 Estimation Error in EBM and EAM

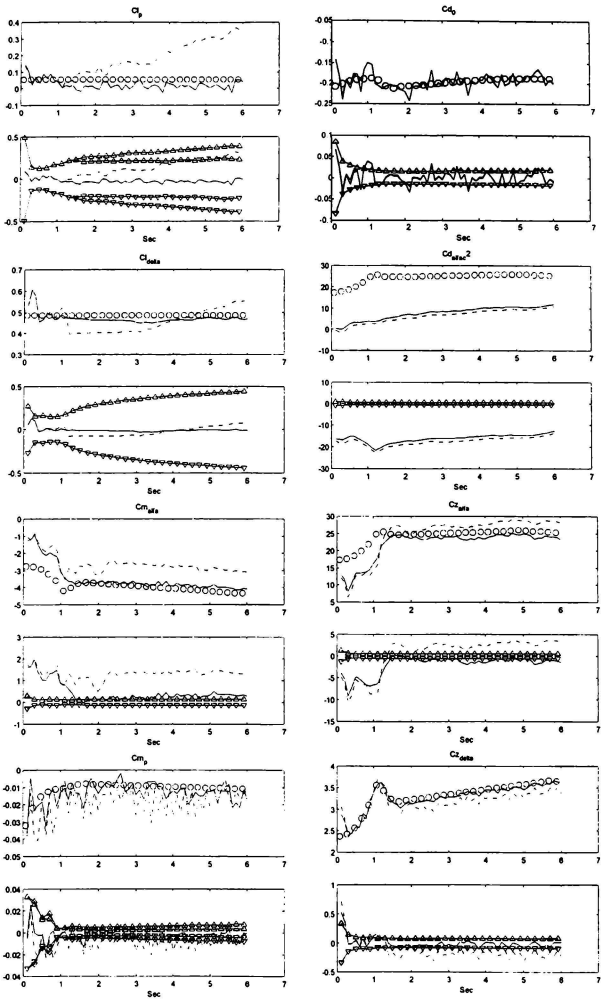


Fig. 5 True Value(o), EKF(-.),EKF_DE(-),Upper and Lower Estimated Error Band(-)

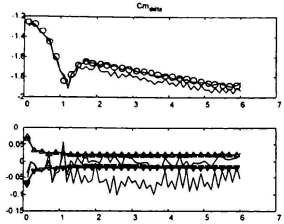


Fig. 5 (Cont.)

References

[1] Mehra R.K. "On the Identification of Variances and Adaptive Kalman Filtering" IEEE Trans. On Automatic Control ,Vol.15, No.2, Apr 1970 , 175-184

[2] Edited by Guarong chen ,Approximate Kalman Filtering, World Scientific, 1993.

[3] Kim K., Shafaei B.,"Rejection disturbances to flexible structure using Pl Kalman Filters" conference on control Applications, Hartford,CT,USA,1997

[4] Linder A.P., Shafaei B. "Robust PFI Kalman Filters", proceedings of the American control Conference, Philadelphia,Pennsylvania, June 1998,pp. 3163-3164

[5] Iliff , K.W. " Parameter Estimation For Flight Vehicles",Journal Of Guidance , Control And Dynamics , Vol.12, No.5 , Sep-Oct 1989 , 609-622

[6] Norton ,F.H. " The Measurement Of Damping In Roll On JN4H In Flight " NACA Rep. 167 , 1923

[7] Norton ,F.H. " A Study Of Longitudinal Dynamic Stability In Flight " NACA Rep. 170 , 1923

[8] Siouris ,G.M. , Chen G. , Wang J.,"Tracking Of Incoming Ballistic Missile Using An Extended Interval Kalman Filter" IEEE Trans. On Aerospace And Electronic Systems, Vol.33, No.1 , Jan 1996 , 232-239

[9] Sri-Jayantha M. , Stengel R.F." Determination Of Nonlinear Aerodynamic Coefficients Using The Estimation-Before-Modeling Method" Journal Of Aircraft, Vol.25 , No.9 , 1988 . 540-542

[10] Jategaonkar,R.V. , Thielecke F. " Evaluation Of Parameter Estimation Methods For Unstable Aircraft" Journal Of Aircraft, Vol.31 , No.3 , 1994 ,540-542

[11] Lee D.M., Lee J. G. " Comparison Of Parameter Identification Algorithms For Flight Vehicles " Trans. Japan Soc. Aero. Space Sci. Vol.36 , No114 , Feb 1994 , 249-256

[12] Jategaonkar,R.V. " Identification Of Actuation System And Aerodynamic Effects Of Direct-Lift-Control Flaps " Journal Of Aircraft, Vol.30 , No.5 , Sep-Oct 1993 ,636-643

[13] Gacia-Velo J. ,Walker B.K. "Aerodynamic Parameter Estimation For High-Performance Aircraft Using Extended Kalman Filtering" ,Journal Of Guidance , Control And Dynamics , Vol.20, No.6 , Dec 1997 ,1257-1259

[14] Jategaonkar,R.V. , Plaettshke "Identification Of Moderately Nonlinear Flight Mechanics System With Additive Process And Measurement Noise " , Journal Of Guidance , Control And Dynamics , Vol.13, No.2 ,Apr-Mar 1990 , 277-285

[15] _____ , "Identification Of Aerodynamic Model Of DLR Research Aircraft ATTAS From Flight Test", Institut Fur Flugmechanik Braunschweig And Sonderforschungsbereich, "Sicherheit In Flugverkehr" Der Technischen Universitat Braunschweig , Sep 1990

[16] Goplaratnam G., Jategaonkar , Some Results Of ATTAS Flight Data Analysis Using Maximum Likelihood Parameter Estimation Method , _____ , Feb 1991

[17] Parameswaran V. , Flight Path Reconstruction Using Extended Kalman Filtering Techniques, _____ , Sep 1990

[18] Preissler H., Schaufele H., "Equation Decoupling - A New Approach To The Aerodynamic Identification Of Unstable Aircraft " , Journal Of Aircraft, Vol.28 , No.2 , Feb 1991 ,146-150

[19] Hall W.E. ,Gupta N.K. "System Identification For Nonlinear Aerodynamic Flight Regimes ", Journal of Spacecraft. Vol.14 , No.2 , Feb 1977 , 73-80

[20] Idan M., "Nonlinear Smoothing Identification Algorithm With Application To Data Consistency Check " Journal Of Guidance , Control And Dynamics , Vol.16, No.2 ,Mar-Apr 1993 , 337-345

[21] Sri-Jayantha M., Data Acquisition And Aerodynamic Coefficient Estimation At High Angles Of Attack , Ph.D. Thesis , 1983 , Princeton University

[22] Chin Ming-Der , Estimation Of Steady And Unsteady Aerodynamic Parameters At High Angles Of Attack , Ph.D. Thesis , 1990 , Texas A&M University

[23] Idan M., Bryson A.E." Parameter Identification Of Linear System Based On Smoothing " Journal Of Guidance , Control And Dynamics , Vol.15, No.4 ,July-Aug 1992

[24] Liu C-Y , Kung M-C , Chen J-C " New Robust And Flexible Parameter Estimation Method " Journal Of Guidance , Control And Dynamics , Vol.16, No.2 , 1992 , 411-413

[25] Shaked U., Desouza C.E." Robust Minimum Variance Filtering" IEEE Trans. On Signal Processing ,Vol.43, No.11, Nov 1995 , 2474-2483

[26] Shaked U.,Berman N., " H∞ Nonlinear Filtering Of Discrete Time Processes" IEEE Trans. On Signal Processing ,Vol.43, No.11, Nov 1995

[27] Nagpal K.M. , Khargonekar P. P. "Filtering and Smoothing in an H∞ Setting" IEEE Trans. On Automatic Control ,Vol.36, No.2, Feb 1991 , 152-166

[28] Singh, Systems & Control Encyclopedia, Theory, Technology, Applications, pp. 4463-4466

[29] Kaminski, P. G. etc. " Discrete Square Root Filtering: A Survey of current Techniques" ,IEEE Trans. On Automatic Control, Vol AC-16, No.6 , Dec.1971, 727-736

[30] P.G.Brown,P.Y.C. Hwang. introduction to random signals and applied kalman filtering,1997, john wiley & sons Inc. , p. 325

[31] David,H.A., Order Statistics ,John Wiley & Sons, NewYork, 1981.

[32] Montogomey, D.C. and E.A. Peck, Introduction to Linear Regression Analysis, John Wiley & Sons, NewYork, 1982.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37250

AIAA-2000-4289

REQUIRED SIMULATION ATTRIBUTES FOR UTILITY*

John Heidt, JECSIM Program Office
Naval Air Warfare Center Weapons Division, China Lake, California
Barry O'Neal, SRS Technologies, Ridgecrest, California

Abstract

Modeling and simulation have long played a significant role in the development, testing, and evaluation of products. However, increasing computing capabilities and decreasing budgets have prompted an increased emphasis on the use of simulations to decrease time cycles, reduce risk, and lower program costs. The fundamental issue that arises in the testing and evaluation of electronic combat systems is the suitability, or utility, of the simulations for an array of typical applications. This paper describes the results of a case study in which Joint Electronic Combat test using SIMulation team members examined the attributes that a simulation application must possess (its utility) to perform successfully four basic test and evaluation functions—planning, prediction, evaluation, and extrapolation.

Introduction

The Joint Electronic Combat test using SIMulation (JECSIM) is a Joint Test and Evaluation (JT&E) program sponsored by the Office of the Secretary of Defense (OSD). JECSIM's charter is to investigate the utility of modeling and simulation (M&S) applications in the test and evaluation (T&E) of electronic countermeasure (ECM) systems.[1] The typical approach in electronic combat (EC) test programs is to test the system (product), fix the product based on the test results, and then test it again, repeating the cycle until the system either passes or fails. However, OSD's current thrust for EC test programs is to supplement testing with M&S techniques and to ensure that testing is conducted only when required.[2,3]

For this case study, JECSIM's approach was to conduct a utility assessment of two ECM devices by utilizing newly developed M&S techniques in a developmental and acceptance (operational) testing process. This assessment was bounded by four basic T&E activities.

1. Planning a test event.
2. Predicting the results of a test event.
3. Evaluating the data collected during a test event.
4. Extrapolating the test event results that pertain to conditions that cannot be tested.

To plan a test event successfully, the simulation must have the ability to qualify the appropriate measure of effectiveness (MOE), recognize the limitations, and determine the requisite instrumentation to collect the test data. Simulations of this nature are built around current knowledge of the mechanics behind similar products developed through the years. The fundamental purpose of this activity is to reduce the risk of test failure.

The second function is characterized by the capacity to predict the results of a measurable event. The primary objective in this effort is to facilitate the determination of an optimal test matrix. At the onset, the tester verifies simple effects and then expands the knowledge base by increasing the complexity in subsequent tests. In the process of capturing results

* This paper is declared a work of the U. S. Government and is not subject to copyright protection in the United States.

Approved for public release; distribution is unlimited.

from prior tests, M&S can sometimes reduce the need to retest and can facilitate the extrapolation to more complex test conditions.

During the evaluation stage, the simulation expedites the resolution of issues that are a consequence of a measurable event. Four fundamental questions arise during this phase:

1. Does the product satisfy the requirements?
2. Does the simulation need to be upgraded?
3. Can the simulation be used to reduce product cost without additional testing?
4. Can the simulation be used to resolve why a test failed?

The simulation's extrapolation role is to predict a non-testable event—one that is too risky or impossible to conduct. This risk may be related to safety, cost, time, or test facility limitations; so testing is often performed with surrogate targets and threats because real ones are unavailable.

The M&S used in this evaluation were built in the Joint Modeling and Simulation System (JMASS) architecture. JMASS, which is an object-oriented environment created to assist designers, engineers, and analysts in the development of digital models, the configuration and execution of simulations, and the analysis of results, consists of a run-time engine that controls the simulation execution and provides a collection of tools that support the configuration and post-processing of the results.

Methodology

The process to assess the simulation's utility consisted of three steps:

1. Define the functional, fidelity, and operational (usability) requirements.
2. Replicate test events with the simulation to produce model prediction and test measurement comparisons.
3. Evaluate the simulation's ability to satisfy the requirement by conducting a survey of a team of M&S users and T&E experts.

Figure 1 depicts a functional framework for the simulation. Egli and Stewart demonstrated this decomposition process in designing the system specifications.[4] The degree of decomposition is a function of the complexity of the simulation or product. Generally, test program managers attempt to minimize the degree of decomposition because the costs associated with verification, validation, and accreditation are directly proportional to it.

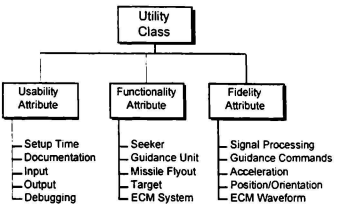


FIGURE 1. A Decomposition of Simulation Attribute.

The methodology for evaluating the simulation utility was based upon examinations of the functional, fidelity, and usability attributes via a requirement scoring process. The investigators devised a four-point scale to represent the simulation's ability to satisfy the requirements, which were derived from the simulation documentation, verification and validation reports, test plans, and experience. Table 1 presents the scores assigned and their significance.

TABLE 1. Evaluation Criteria for Attribute Requirements.

Score	Functionality	Fidelity	Usability
0	Requirement not met (function not implemented in model)	No correlation exists	Impossible to do
1	Requirement somewhat met (simple or limited implementation)	Poor correlation (<0.5) exists	Difficult to do, but possible
2	Requirement mostly (>50%) met (analytic function)	Correlation could be better	Possible to do, but not easily
3	Requirement met or exceeded (data lookup table)	Perfect correlation (or as good as it gets) exists	Easy to do

The investigators defined functional requirements deemed essential in predicting measures that would be evaluated during testing. Each component model in the simulation (i.e., the seeker, the missile, the target, the EC system) was evaluated with respect to these functional requirements. This analysis was analogous to a verification activity, but the software itself (i.e., code) was not examined. Then, the investigators developed the functional scores from interviews with the model developer and from a review of the available documentation.

Next, the investigators determined the fidelity requirements for each of the test measures of performance (MOPs) that the simulation would predict and evaluated these requirements for a specific type of test activity. The scores were derived from comparisons between the M&S forecasts and the test data for applicable models and types of testing. When predicting MOPs for a hardware-in-the-loop test, the investigators examined only the seeker model. However, the predictions for a live-fire test involved evaluations of both seeker and missile flyout models, which were equally weighted and combined.

The investigators evaluated the integrated simulation by assessing usability requirements that addressed such issues as adequate documentation, satisfactory support, and viable error condition reporting. The JECSIM team members who were responsible for using the simulation to make predictions for each of the utility evaluations provided the scores.

Each expert was also asked to weight the importance of the requirements relative to each of the T&E applications (see Table 2). A limitation of this methodology was the selection of weighting factors derived from a small sample size, which was statistically insignificant. However, the small range of variability in the responses indicated those attributes deemed important in satisfying utility requirements.[5]

TABLE 2. Importance Weighting Factors.

	Functionality	Fidelity	Usability	Total
Planning	3.5	2.7	3.8	10
Prediction	3.4	4.3	2.3	10
Evaluation	3.9	3.7	2.4	10
Extrapolation	4.0	4.7	1.3	10

These weighting factors were used in combining the functional, fidelity, and usability scores (Equation 1).

$$U = \sum(U_i \times Wu_i + F_i \times WF_i + Fid_i \times WFiD_i) \quad (1)$$

where:

U = usability score
Wu = usability importance weighting factor
F = functional score
WF = functional importance weighting factor
Fid = fidelity score
WFiD = fidelity importance weighting factor

Then, the investigators normalized the weighting factors for each application (Equation 2).

$$Wx = Wi/3.333 \quad (2)$$

where:

Wx = weighting factor for application score *Wi*
Wi = relative importance of attribute *i*

The resulting utility score was a value between 0 and 9. This scheme was designed to facilitate a qualitative evaluation of the simulation via an accepted methodology. Figure 2 incorporates a logic flowchart process described by Rival and Comfort[6] for the subjective evaluation of military systems.

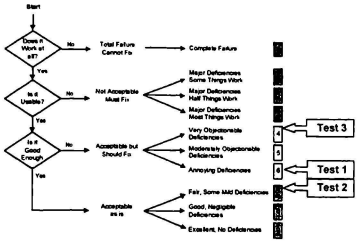


FIGURE 2. Meaning of Utility Scores.

The investigators assessed the status of the simulation by plotting the attribute utility scores according to the scale on the right side of the flowchart—the severity of the deficiencies correspond to this score. A typical utilization for this type of scoring system is the evaluation of the handling qualities of an aircraft. In this kind of application, factors that vary in importance among users are as relevant (in some cases) as the technical properties of the system. As our M&S evaluators demonstrated, most users consider predictive accuracy more important than ease-of-use factors, but the relative importance varies across applications.

Results

Figure 3 depicts the weighting factors and shows that, for functionality, they were approximately the same (one third of the total) across all applications. The evaluators preferred increased fidelity over ease of model use as the applications became more predictive and critical (cost effective) in nature. This finding is not surprising because the product developer's reason for utilizing M&S is to reduce the risk of failure, which increases as the product moves further away from the current knowledge base. Therefore, ease-of-use issues become less important. The lack of variability among the evaluators is also worth noting. Even though these personnel represented a diverse group in terms of experience and background, their willingness to sacrifice ease of use for predictive accuracy was unanimous.

Figure 4 illustrates the attribute scores and the resulting utility score, for an evaluation. The bar charts show the percentage of 0, 1, 2, or 3 scores (from left to right) received during the requirement evaluations. In this example, about 80% of the functional requirements were satisfied. However, fidelity and usability scores were spread across the spectrum, while some requirements were not met. The utility score (6.5) was computed by averaging the attribute scores for each requirement weighting and then adding the three results together (Equation 1). The composite score indicated deficiencies (from the Figure 2 flowchart) between annoying and mild. Even though the evaluation process was subjective, sampling a spectrum of users and considering a scope of applications (via the weighting factors) yielded a quantitative measure that can be used to compare various simulations and evaluate improvements in a simulation over time.

A key finding was that developing requirements for the evaluation of a simulation is a relatively simple task once its intended uses and the required test measures have been defined. The investigators also determined that the functionality attribute can be used to aid in the selection and development of a simulation for use in the evaluation of a product.

In this case study, while the functional scores were high, those for fidelity varied across the model-to-test comparisons according to the type of test data and application. Combining the scores from evaluations that spanned different test and functional situations made it possible to color code the utility for an application (with the Figure 2 flowchart as a guide). A stoplight (i.e., red, yellow, and green) chart that illustrates this type of evaluation result is depicted in Figure 5; and it suggests adequate utility for some cases but not for others. Moreover, this simple technique provides the engineer, manager, and analyst an easy way to communicate the strengths and weaknesses of a simulation.

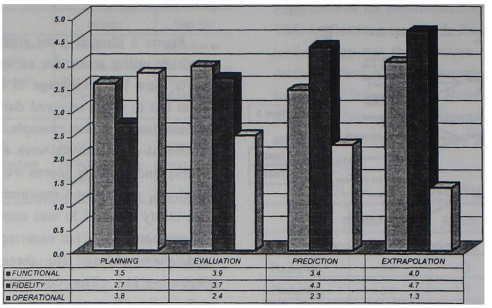


FIGURE 3. Requirement Weighting Factors for Applications.

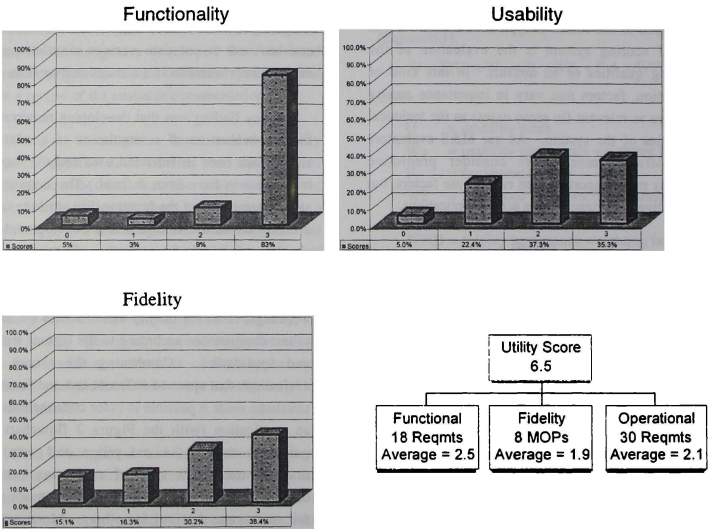


FIGURE 4. Attribute and Utility Scores for a Prediction Application.

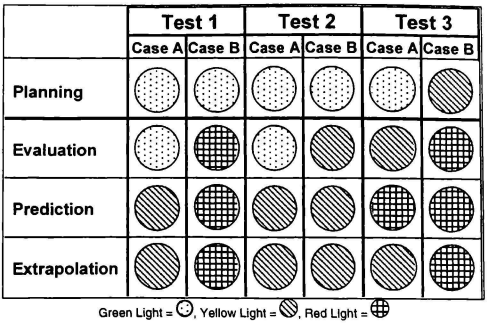


FIGURE 5. Simulation Utility Across Test Cases and Applications.

Even though the simulation exhibited fidelity problems that limited its utility for some of the relatively critical functions, its effectiveness in the planning stage was adequate because the need for fidelity during this stage was not deemed essential. In this phase, managers use simulations to allocate resources and avoid poor MOEs for upcoming test events. Besides, utility is strongly related to usability factors that facilitate running the simulation quickly so that every possible outcome is evaluated. Figure 6 shows the percentage of MOEs that were dropped or modified during the testing process when personnel utilized simulations to analyze the planned test measures. These percentages are normalized to the initial number of MOEs.

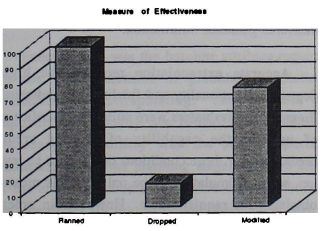


FIGURE 6. MOE Modification in Test Planning.

The effect of the simulation on the reduction and refinement of the evaluation measures for this test activity illustrates an improvement to the model–test–model process in decreasing time cycles. In addition, the cost of pre-test analysis is somewhat offset by what would have been incurred post-test for analyzing unnecessary measures. Moreover, performing pre-test simulation analyses offers other potential benefits. First, the simulation results can be used to communicate expectations to the test team, thereby reducing the risk of test failure. Second, these analyses also help the testers understand what is and is not known about the devices to be tested. Third, the simulation can be used in sensitivity analyses to examine and help define data resolution, range, and accuracy requirements. These analyses can result in important implications for test instrumentation and, therefore, facility requirements.

Figure 7 illustrates the impact that both negative and positive usability attributes can create. Two different simulation versions were utilized to produce comparisons for a test activity. Because the second version had a significantly greater automation capability and better documentation, the users produced 35 times more runs over an equivalent time period. Even though the cost of repeating the analysis and the evaluation was not planned, the

improved usability of the new version allowed the project personnel to recover lost time.

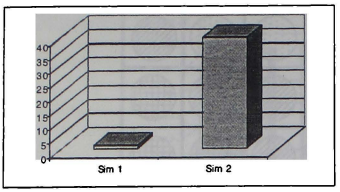


FIGURE 7. Number of Runs Completed by Two Versions.

The evaluators indicated that functional and fidelity requirements were the primary factors for prediction and evaluation. One of the key drivers for functional requisites was the simulation's ability to produce the results in various formats because, for utility assessments, test data are provided in different configurations and coordinate systems. Thus, in order to present these results to the experts, JECSIM had to translate the outcomes into the pertinent formats off-line. Therefore, the inability of the simulation to report the data in units common with those used by the test activity created a situation in which additional resources had to be allocated to achieve that objective.

The simulation's credibility was driven by its ability to replicate accurately test data at both system and subsystem levels. Figure 8 shows the difference in the predictive results for two tests of a common scenario made by two separate runs. The simulation was designed to represent the performance capability of the system for ECM 1 but had not been validated for the conditions of ECM 2. The chart shows variances, which are modest for SIM 1 but significant for the SIM 2, between the predictions and the actual test measurements for four cases. In other words, valid conclusions would have been drawn from the simulation predictions for ECM 1 but not for ECM 2. As is often the case, the cause of this discrepancy was traced to a bug in the implementation of a subsystem that ultimately affected the system performance predictions. However, debugging was simple because the correlation of the predicted and the measured subsystem responses was also poor (i.e., received low fidelity evaluation scores).

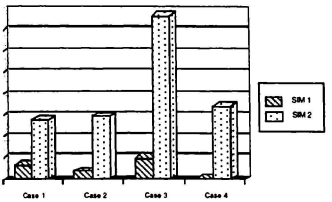


FIGURE 8. Differences Between Test and Simulation Results. (Each bar reflects the variance between the actual test results and the simulation results.).

A key usability subattribute for evaluation was the ability to inject measured test data into the simulation. This capability provides a convenient mechanism for adapting a simulation to evaluate higher-level effects that are difficult to measure. A common example is utilizing measured guidance commands to drive a missile flyout simulation to produce miss distance data. When a simulation is applied in this manner, low-level software defects (such as the one that influenced the Figure 8 outcomes) can be bypassed and other parts of the simulation can be isolated for careful examination. Before the results of such an evaluation can be applied, however, the simulation must be validated for the intended use. In other words, missile flyout models must be validated for the purposes of predicting miss distances even when incorporating measured guidance information.

Accurate extrapolation requires the highest degree of functionality and fidelity. As indicated in Figure 3, the evaluators were willing to sacrifice ease of use for the capability of making credible predictions for those conditions that cannot be tested, i.e., when a simulation is the only feasible means of obtaining the requisite data. However, in order to ensure an accurate prediction of other test case results, the simulation must first be validated against known outcomes. Then, valid representations of

other systems (targets, seekers, or missiles) incorporate these data to evaluate the consequent effects. Again, as the reader can see, the ability to inject data into or use data to drive the simulation becomes critical for this function. However, the validation of other system representations is likely to be incomplete or subjective; so these limitations must be considered when applying the results.

For example, JECSIM evaluated the effects of different seekers (variants) on simulation predictions and discovered important performance relationships among the internal subsystems. However, these results could not be extended to system-level measures because of the low credibility exhibited in portions of the missile flyout simulation. Therefore, the extrapolation application must be approached with caution until critical, if not all, parts of the simulation have been thoroughly tested and validated. When the utility of this application has been achieved, M&S will have much to offer the T&E process.

Conclusions

The JECSIM team evaluated a simulation's utility as a function of three fundamental attributes: functionality, fidelity, and usability. The personnel found that the relative importance of these attributes was application dependent and that the simulation should facilitate discrimination among test cases for common T&E uses. In addition, the study demonstrated the feasibility of adopting a systematic approach to the evaluation of M&S utility for the development and acceptance testing of a product.

Functionality was the baseline attribute required for the T&E functions of planning, predicting, evaluation, and extrapolating. Thus, without the requisite functionality, no further evaluation was necessary. Usability was the most important characteristic for the planning stage, in which ease-of-use and production factors were critical. Fidelity was the driver for the prediction phase, especially in those cases in which M&S was used to supplement testing or reduce the requirements. This attribute was also significant in evaluation and extrapolation. As a matter of fact, in these functions, usability became even less important and the desire for fidelity increased even more.

The ability to inject test data into models and simulations proved to be a key concern for the most critical applications—evaluation and extrapolation. However, because this requirement was not addressed during the initial development, this feature proved difficult to achieve. Moreover, in order to use simulations to extend or extrapolate test measurements, component model validation must be accomplished. Even though, in this study, the seeker model was detailed enough to characterize differences that were important in the evaluation of ECM technique robustness, an inability to validate the missile flyout portion of the simulation precluded an evaluation of seeker variability effects on higher-level measures or for other (untested) cases. In addition, complex simulations require the characterization and validation of the submodels (components) in order to build credibility and impart utility for T&E.

An unexpected, but consequential, finding was that the results encountered by the 6-member JECSIM evaluation team were not significantly changed by the incorporation of inputs from 26 additional people. Figure 9 illustrates the similarity of the outcomes for a typical fidelity evaluation from both 6 and 26 participants.

When evaluations like this one are conducted, a relatively small cross-functional team composed of experts in various aspects of a product or system can be as effective as a larger, more diverse cross-section of contributors. Also, when appraising complex phenomena over a wide array of conditions, repeated exposure and learning facilitated the discrimination of important details; and, thus, the assessments became more meaningful. Many of the one-time evaluators were unable to score some of the requirements, but those who participated more than once reported an improved ability to make qualified decisions about functionality and fidelity issues.

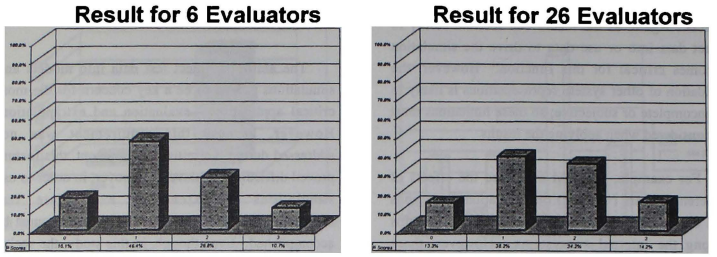


FIGURE 9. Effect of Evaluation Team Size on Results.

The results of the JECSIM case study proved that the utility of M&S could be quantified in terms of three attributes (functionality, fidelity, and usability) that were based upon many lower-level requirements. The importance of these attributes across the types of applications required to support T&E activities facilitated the discrimination of the utility among them for different types of tests. The utility scores supported the limitations identified related to specific test applications and served to highlight areas in which further developments and improvements were needed. In other words, the methodology was successful in making the results of an inherently subjective evaluation more objective and relevant to the M&S user. Finally, the repeated examinations of the simulation results in light of the requirements allowed a small team of experienced evaluators to arrive at the same conclusions reached by a much larger, more statistically significant sample size, a quality that makes the methodology more manageable and cost effective.

References

1. *Program Test Plan for the Joint Electronic Combat test using SIMulation Joint Test and Evaluation*, by Mike Franklin, et al, 28 October 1998. (Report number M00242, publication UNCLASSIFIED.)

2. Capt. Pierre G. Vining. "Operational Tester Reluctance To Use Modeling and Simulation," *ITEA Journal*, December 1998, pp. 13-17.

3. Lorraine M. Shea. "Modeling and Simulation: Striking a Balance in Navy Resource Allocation," *ITEA Journal*, December 1998, pp. 18-20.

4. R. Egli and N. F. Stewart. "A Framework for System Specification Using Chains on Cell Complexes," in *Computer-aided Design, Vol. 31*, 1999, pp. 669-681.

5. *Discrete-event Simulation Software: What the Users Want, Simulation 73.6*, by Vlatka Hlupic, 1999, pp 362-370.

6. Institute for Defense Analysis. *A Quantitative Approach to Qualitative Measures of Performance*, by A. Rex Rival and Gary Comfort, IDA, December 1996. (Document D-1952, publication UNCLASSIFIED.)

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37251

AIAA-2000-4290

VARIABLE WINDOWS AND FORCING FUNCTION ALGORITHMS FOR HARDWARE IN THE LOOP SIMULATIONS

Richard Gorecki*[1]
Defence Science and Technology Organisation,
Salisbury, South Australia, 5108

Abstract

The DSTO Hardware in the Loop (HIL) facility, currently has a limited Radio Frequency Scene Generator (RFSG) that restricts the scope of HIL simulations. Furthermore it is a multi-use facility and hence has restricted availability. This report derives algorithms and methods and indicates ways that can enable HIL simulations of most real world engagement scenarios. The variable window algorithm is the key algorithm and when used to expand simulations must be complemented by the injection of a forcing function into the missile tracking loop. By adding a zero motion compensation algorithm closed loop HIL simulations can be run without the need to use the motion table and the RFSG. The algorithms and method have been successfully applied and used in the HIL facility.

1. INTRODUCTION

The work was done with a view to developing for the Hardware in the Loop (HIL) installation a generic capability to significantly expand the scope of HIL simulations. At this time the main limitation stems from the relatively small angular extent of the Radio Frequency Scene Generator (RFSG) horn array. So, if the facility is used in the basic way then the scenario is restricted to head-on or tail-on engagements against low crossing rate, non-manoeuvring targets. Furthermore glint, clutter, and multipath or multiple target effects may not be possible or only just so providing the experiment is carefully designed or the antenna of the unit under test has a narrow beam-width.

The ideas for the capability are based on the work of Lange[1] and Snyder[2,3]. Unfortunately the detail of the algorithms could not be obtained for various reasons, so it was necessary to develop our own algorithms. This report describes the algorithms and the underlying concepts that in hindsight are simple.

The HIL facility in DSTO is a multi-use one for carrying out testing and trials on a range of systems and missiles. This could lead to problems in time sharing of the facility in the development phase of HIL simulations particularly in the integration of the hardware to be tested with the software part of the simulation. Missile HIL simulations can only be fully verified by running closed-loop homing simulations and this can also be a time consuming process requiring the use of the full facility. In order to use the facility more efficiently and to allow more time for development of HIL simulations the variable window and forcing function concept has been extended to allow closed loop homing simulations using a fixed target source and static missile hardware.

2. HARDWARE AND SOFTWARE SPACE

In a HIL simulation of a missile engagement the software generates commands, based on geometry, for the motion table and scene generator thus providing stimulation of the missile seeker which generates commands to drive the software missile and hence change the geometry. It is easier to consider the detail of this complex interaction between hardware and software by using the concepts of hardware and software space.

Thus software space is concerned with the missile autopilot, airframe motion, target motion as well as glint, clutter, multipath and environment effects. Software space produces the relative geometry of the engagement. Hardware space contains the motion table, a scene generator ie RF, IR or Optical, and the missile seeker and produces guidance commands. Each space has a reference frame that ties all the separate frames within it together. These are the Earth frame in software space and the HIL reference frame in hardware space. The HIL reference frame is the overall reference.

In a basic HIL simulation making the Earth frame coincident with the HIL reference frame links the spaces. In this case the scope of the simulation is limited by the hardware limitations. The total LOS rotation is confined to the angular dimensions of the horn array (RF) or outer table gimbal limits (IR). The missile angular motion comprising body plus manoeuvre is restricted by the table gimbal limits. In this facility the RFSG horn array is about the same size as the antenna beamwidth of many RF missiles,

[1] *Senior Research Scientist

Copyright © 2000 by R. Gorecki. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

which makes it the dominant limiter. However this size limitation can be overcome by confining the LOS to the centre of the scene generator. Linking the spaces with a variable window frame can do this.

3. VARIABLE WINDOWS AND FORCING FUNCTIONS

The variable window is basically a field of view, having the angular extent of the scene generator, which is viewed from the missile reference frame origin. The field of view is about a bore-sight that can be aligned with the OX axis of any existing axis system, or one defined for the purpose, in software space. The main requirement is that this field of view or window contains the relevant targets for the duration of the engagement. A variable window frame is then defined. Aligning the variable window frame with the earth frame gives the basic configuration. However, if the variable window frame is aligned with the missile to target LOS frame then the target will always be in the centre of the window, and by this linking process, the centre of the scene generator. It is convenient to introduce the concept of a virtual seeker in the software space that is linked to the actual seeker in hardware space. In the basic configuration the simulated target moves about so the seeker hardware stimulation geometry follows the virtual seeker stimulation geometry and the measured bore-sight error history is approximately the same as the virtual seeker geometric bore-sight error history. However in the other configuration above, for the same engagement conditions, the hardware stimulation geometry does not change so after possibly an initial transient the bore-sight errors are zero and the history is not preserved. Since in air to air missiles the basic measurement from which the guidance commands are derived is the bore-sight error then the missile trajectory will not be preserved and the simulated engagement will fail.
So, for this method to work the motion table and scene generator demands must be relative to the variable window frame but more crucially the tracker must be forced to generate bore-sight error histories identical to ones for the same engagement conditions using the basic configuration. At this stage this forcing can only be applied to a certain type of tracker. The tracker needs to be space stabilised ie it automatically compensates for missile body motion in which case the head drive commands are estimates of the LOS rate and they cause the head to track the LOS in inertial space (earth frame). So the desired situation is achieved if the forcing functions remove appropriately resolved components of the variable frame rotation rate from the head drive commands. Thus if the basic configuration is used the variable frame rate is zero ie the earth frame rate, and the head drive commands are unaltered. If the other configuration discussed above is used then the variable frame rate is the LOS rate and forcing functions modify the head drive commands so that they are the negative of the bore-sight error rates. This produces the correct bore-sight error history. For any other variable frame configuration the head movement will also result in the correct error history. Note that in practice the dynamic responses of the hardware involved will influence these results. This will be discussed further in Appendix A.

4. ZERO MOTION CLOSED LOOP HIL SIMULATIONS

The technique of using variable windows and forcing function injection has been extended to allow closed loop homing simulations without a motion table. The need for this arose from the limited availability of the HIL facility resources. At this stage it is a multi use facility under development thus restricting availability to each project to a fixed time window of limited duration. From experience, integrating and verifying such a complex process as a HIL simulation can take much time apart from manpower resources required for staffing the full facility. There is also the possibility of damage to the unit from jolting in uncontrolled motion. All these factors drove the development of algorithms that enable the integration of the unit hardware with the real time computer and the verification of the simulation with a static rig and a single source.
This method is not intended for realistic simulations because it does not produce the correct target radiation paths through the missile nose cone. The method assumes a single point source target and aligns the variable window frame with the LOS frame thus ensuring a fixed target source. As before, forcing function signals need to be injected into the seeker. To compensate for the static missile hardware the guidance commands to the autopilot need to be modified to reflect the correct relativity between the detector bore-sight and the missile airframe.

5. DIGITAL AND HIL SIMULATION RESULTS

The HIL system model[4] shown in figure 1 has been combined with a generic RF missile simulation model that includes a model of the LOS rate estimation process. This has been used to test the variable window frame implementation. It was assumed that the HIL motion table gimbals had a simple lag response of .01 s and that the RFSG response was delayed by 4 ms. There was no parallax.

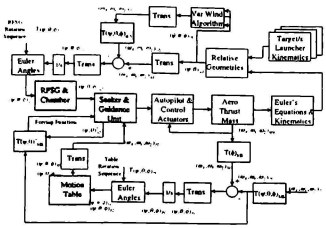


Figure 1. Block diagram of HIL simulation with variable windows and forcing functions.

The basic experiment was to introduce a step in azimuth LOS rate of approximately 3 deg/s by firing the missile at a crossing target. At the same time there was a step of about .5 deg/s in elevation due to an altitude separation.

Four cases were looked at in which the seeker estimates, $(\dot{\omega}_{yLD}, \dot{\omega}_{zLD})$, of the LOS elevation and azimuth rate as well as bore-sight error histories were compared. Also plotted were the software space LOS elevation and azimuth rate components in detector axes, $(\omega_{yLD}, \omega_{zLD})$, and the forcing function or variable window frame azimuth and elevation rate components in detector axes, $(\omega_{yVD}, \omega_{zVD})$. The forcing functions were summed with the detector head rate commands. The HIL LOS rate components are the hardware space rates between the virtual source and the detector.

The cases are compared with a simulation of the missile only.

The variable window frame was aligned with the

(a) Earth frame, ie $\dot{\psi}_{VE} = 0.0$ and $\dot{\theta}_{VE} = 0.0$. This is the basic configuration.

(b) LOS frame, ie $\dot{\psi}_{VE} = \dot{\psi}_{LE}$ and $\dot{\theta}_{VE} = \dot{\theta}_{LE}$. This is typical usage of the variable window technique.

(c) LOS frame plus an additional sinusoidal component of azimuth rate. ie

$\dot{\psi}_{VE} = \dot{\psi}_{LE} + 0.5\sin 10t$ and $\dot{\theta}_{VE} = \dot{\theta}_{LE}$

(d) LOS frame, ie $\dot{\psi}_{VE} = \dot{\psi}_{LE}$ and $\dot{\theta}_{VE} = \dot{\theta}_{LE}$. This case is for zero motion closed loop simulation.

Case (c) illustrates the idea of using the LOS to a centroid of 2 targets.

Figures 2 to 7 show results from the experiments. In each case the missile guided to a successful intercept. The actual LOS rates, which are precise indicators of rate of change of relative geometry, are used for comparison. A repeated LOS rate history indicates a repeated bore-sight error history since the relative geometry results from the missile trajectory that results from the guidance commands that are derived from the bore-sight errors. Bore-sight error histories are included in the figures.

Figure 2 shows the results from the base run.

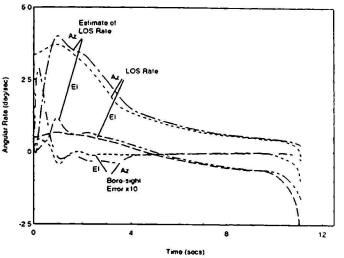


Figure 2. LOS rate components in detector axes missile only.

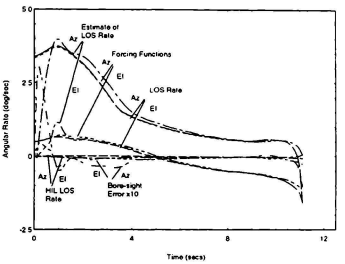


Figure 3. Case (a): LOS rate components in detector axes: variable frame equals earth frame.

Figures 3 and 4 show that the engagement instantaneous geometry for case (b) is virtually unchanged, apart for initial transients, compared with the base case (a). Note that the HIL LOS rate differs slightly from the actual LOS rate in case (a) and is not quite zero in cases (b) and (c). This is due to the lags and delays in the HIL systems.

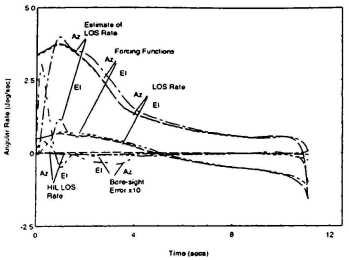


Figure 4. Case (b): LOS rate components in detector axes: variable frame equals LOS frame.

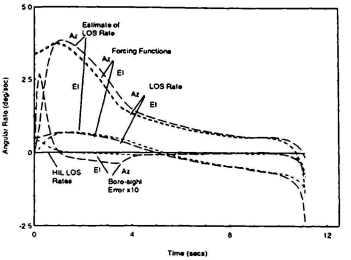


Figure 6. Case (d): LOS rate components in detector axes: variable frame equals LOS frame and zero motion.

Figure 5 shows that basically the same bore-sight error histories ie the same relationship between the detector bore-sight and the HIL LOS are maintained despite the modulation. This is shown up particularly by the fact that the modulation on the estimate of the LOS is quite small, eg a 2 deg/s modulation on the variable frame rate becomes a 0.06 deg/s modulation on the LOS estimate ie a 33 to 1 reduction.

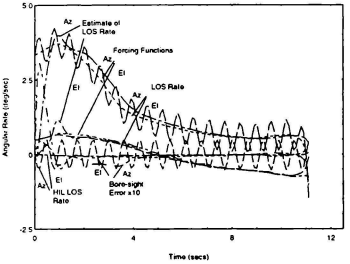


Figure 5. Case (c): LOS rate components in detector axes: variable frame equals LOS frame plus modulation.

Figure 6 shows that the LOS estimates for the zero motion case are smoother in the first two or so seconds. This is probably due to the fact that the tracking head is operating in a linear region around its zero operating point since it is not compensating for missile body rotations. The head angles are approximately the negative of the bore-sight errors vis, less than 0.25 degrees.

Figure 7 compares seeker estimates of the LOS rate components for the 4 cases. Using case (a) as the base they show that the estimates compare reasonably well at first but diverge towards the end of the simulation. This is mainly due to the seeker estimation process that uses Kalman filters, which have history dependent gains. So errors (eg initialisation) have a cumulative effect on the engagement

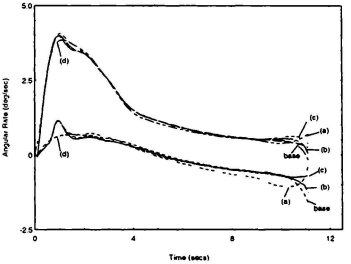


Figure 7 Comparison of estimated LOS rate azimuth components.

6. CONCLUSION

This paper has outlined a method of implementing variable window frames in HIL simulations. This requires the injection of forcing functions into the missile hardware. The method has been extended to allow closed loop homing simulations with a static rig and a single source.

Although the generation of the motion table demands and the scene generator demands may be generic the forcing functions may need to be tailored to the missile injection point as well as requiring head angle information from the missile.

The algorithms to control the orientation of the variable window frame relative to the local earth frame will need to be developed. The algorithms may be developed to allow multiple target scenarios (see Appendix A) to be presented. In such a case, it would be advantageous to have access to head pointing information from the missile seeker hardware.
The idea, mentioned in Reference 1, that compensatory terms can be included in the forcing functions that virtually eliminate parallax will need to be investigated, as well an alternative idea of including parallax correction in the angle of arrival demands.

6. REFERENCES

1. Lange A.S,"Semi-Physical Simulation of Guided Missiles", Comput. & Elect. Engng, Vol.1 pp. 119-142 Pergamon Press, 1973.
2. Snyder H.P.,"Artificially Generating Line-of-Sight Rates For Hardware-in-the-Loop, Missile Interceptor Simulations" Proceedings of the 19?? Summer Simulation Conference
3. Snyder H.P,"Improved Multiple Target Capability for a Guidance Test Simulation Facility", Appendices Raytheon Co., Bedford, Mass,M/S M3-32
4. Gorecki R.M., "A Mathematical Model for Simulating a Hardware in the Loop Facility". DSTO-TR-0931, March 2000.
5. Gorecki R.M., "A Baseline 6 DOF Mathematical Model of a Generic Missile". DSTO-TR-0930, March 2000.
6. Nesline F.W. and Zarchan P.,"Line-of-Sight Reconstruction for Faster Homing Guidance". AIAA Guidance and Control Conference, August 1983, Volume 8, No. 1 (Jan-Feb)

7. APPENDIX

A.1. Subscripting Convention

The following rules apply to the subscripting convention used in this report. When considering a property such as angular rate of an entity, the first upper case subscript refers to the entity and the second upper case subscript refers to the coordinate system in which the components of the property are measured. When considering rotations or transformations between coordinate systems the first upper case subscript refers to the rotated coordinate system and the second upper case subscript refers to the reference coordinate system. The orientation of the rotated coordinate system with respect to the reference coordinate system is defined by Euler angle rotations. The entities and their coordinate systems that are of interest in this report are described in the following section.

A.2. Coordinate Systems

Reference 5 gives definitions for coordinate systems used in digital simulations of missile engagements. The coordinate systems use the right-handed axis convention with the X axis pointing forward, if the Z axis is downward, then the Y axis points to the right when looking along the X axis. The coordinate systems will be generally referred to as frames.

A.2.1 Local Earth coordinate system $(X,Y,Z)_E$

The direction of the X_E-axis is chosen as the down-range direction and origin coincides with the projection of the initial target position on the ground. The Z_E-axis is vertically downwards and the Y_E-axis is normal to the XOZ plane and to the right. Relative kinematics are calculated in a moving earth coordinate system whose origin coincides with the centre of gravity of the reference body, eg the target position relative to the missile is calculated in an earth coordinate system attached to the missile.

A.2.2 Missile body coordinate system $(X,Y,Z)_M$

The origin is located at the missile centre of gravity and the axes are fixed relative to the missile body. The X_M-axis is along the centre line of the missile body and positive forward. The Y_M-axis is perpendicular to the X_M-axis and contained in the plane of the undeflected pitch motivators. It is positive to the right. The Z_M-axis is mutually perpendicular to the X_M and Y_M axes and contained in the plane of the undeflected yaw motivators. If Y_M is horizontal Z_M is positive downwards.

A.2.3 Missile seeker reference coordinate system $(X,Y,Z)_S$;software space

The origin is located at the missile centre of gravity and the axes are fixed relative to the missile body. The X_S-axis is along the centre line of the missile body and positive forward. The Z_S-axis is perpendicular to the X_S-axis and contained in the plane defined by the X_S-axis and the centred azimuth gimbal. It is positive down. The Y_S-axis is mutually perpendicular to the X_S and Z_S axes and positive to the right.

A.2.4 Missile unit hardware seeker reference coordinate system $(X,Y,Z)_U$;hardware space

The origin is made coincident with the payload reference system origin and the axes are fixed relative to the missile body. The X_U-axis is along the centre line of the unit and positive forward. The Z_U-axis is perpendicular to the X_U-axis and contained in the plane defined by the X_U-axis and the centred azimuth

gimbal. It is positive down. The Y_U-axis is mutually perpendicular to the X_U and Z_U axes and positive to the right.

A.2.5 Line-of-Sight coordinate system $(X,Y,Z)_L$

The origin is located at the missile centre of gravity. The X_L-axis is aligned with the missile to target relative range vector. The Euler rotations psi and theta (no rotation about the OX axis) define the XL-axis direction relative to the moving earth coordinate system. The Y_L axis is horizontal and to the right of OX and the Z_L axis completes the right-handed set.

Variable window coordinate system $(X,Y,Z)_V$

The X_V-axis is aligned according to some function that depends on the sightlines to one or more targets. The origin is coincident with the centre of gravity of the missile. The Y_V and Z_V axes complete a right-handed set. Their directions may also be dependent on the scene generator configuration and the type of target and its representation. This system is defined in software space and links it to the hardware space by superimposing it on the HIL reference coordinate system.

A.2.7 HIL reference coordinate system $(X,Y,Z)_R$

Ideally the axes are aligned with the table gimbals in their zero position where the three table axes are orthogonal. The origin is coincident with the centre of rotation of the table. The X_R-axis is along the line joining the origin to the centre of the scene generator. Ideally it is aligned with the motion table roll gimbal The Y_R axis is horizontal and to the right and the Z_R axis completes a right-handed set. They may be aligned with either of the other gimbals depending on the location and orientation of the scene and the requirements for missile hardware movement.

A.2.8 Scene Generator coordinate system $(X,Y,Z)_C$

This system is aligned and coincident with the HIL reference coordinate system. It is assumed that the scene generator array or range of motion is confined to a spherical surface whose centre coincides with the system origin.

A.2.9 Table Payload Reference coordinate system$(X,Y,Z)_P$

This is a right-handed system whose axes are aligned with the table gimbals in their zero position where they are orthogonal. The origin is coincident with the centre of rotation of the table. This system conforms to the convention adopted by the manufacturer for the table control. For example when the table is used with the Radio Frequency Scene Generator (RFSG) it is rotated 180 degrees anticlockwise about the X_R axis with respect to the HIL reference coordinate system. When the unit is supported by a static rig it's relationship to the HIL reference is assumed to be the same as the motion table payload system.

A.3. Outputs to Motion Table

This type of derivation has been given in other documents (eg reference 3) but the detail is repeated here for convenience. To simply illustrate the algorithm it is assumed that the Table Payload reference frame is aligned with the HIL reference, Scene generator and Variable Window frames.

A.3.1 Missile body attitude and angular rates relative to the variable window frame

Since the motion table holds the seeker section, the seeker reference frame is used as the reference frame for the missile and hence the Euler rates $(\dot{\psi},\dot{\theta},\dot{\phi})_{sv}$ need to be calculated.

The variable window rate $\overline{\omega}_v$ can be defined by

$$\overline{\omega}_v = \omega_{xvv}\hat{i}_v + \omega_{yvv}\hat{j}_v + \omega_{zvv}\hat{k}_v$$

Where $\omega_{xvv}, \omega_{yvv}, \omega_{zvv}$ are the magnitudes of the rates about the OX_V, OY_V and OZ_V axes respectively and $\hat{i}_v, \hat{j}_v, \hat{k}_v$ are unit vectors along these spin axes.

The seeker reference frame rate $\overline{\omega}_s$ can be defined by

$$\overline{\omega}_s = \omega_{xss}\hat{i}_s + \omega_{yss}\hat{j}_s + \omega_{zss}\hat{k}_s$$

Where $\omega_{xss}, \omega_{yss}, \omega_{zss}$ are the magnitudes of the rates about the OX_S, OY_S and OZ_S axes respectively and $\hat{i}_s, \hat{j}_s, \hat{k}_s$ are unit vectors along these spin axes.

The seeker reference frame rate relative to the variable window frame rate is

$$\overline{\omega}_{sv} = \overline{\omega}_s - \overline{\omega}_v$$

We can also define $\overline{\omega}_{sv}$ in terms of the Euler rates

$$\overline{\omega}_{sv} = \dot{\phi}_{sv}\hat{i}_s + \dot{\theta}_{sv}\hat{j}_v + \dot{\psi}_{sv}\hat{k}_v$$

Where $\hat{i}_s, \hat{j}_v, \hat{k}_v$ are unit vectors along the spin axes of the rates having magnitudes $\dot{\phi}_{sv}, \dot{\theta}_{sv} and \dot{\psi}_{sv}$.

The components of the above vectors must be in the same frame and it is convenient to use the seeker reference frame as the common frame.

So

$$\begin{pmatrix} \omega_{xvs} \\ \omega_{yvs} \\ \omega_{zvs} \end{pmatrix} = \mathbf{T}(\dot{\psi},\dot{\theta},\dot{\phi})_{sv} \begin{pmatrix} \omega_{xvv} \\ \omega_{yvv} \\ \omega_{zvv} \end{pmatrix}$$

and

$$\overline{\omega}_{sv} = (\omega_{xss} - \omega_{xvs})\hat{i}_s + (\omega_{yss} - \omega_{yvs})\hat{j}_s + (\omega_{zss} - \omega_{zvs})\hat{k}_s$$

$$- \omega_{xvv}\hat{i}_v + \omega_{yvv}\hat{j}_v + \omega_{zvv}\hat{k}_v$$

Also the Euler angle rates are resolved into seeker reference axes as follows

$$\bar{\omega}_{sv} = \dot{\phi}_{sv}\bar{i}_s + \left[\mathbf{R}_1(\phi_{sv})\begin{pmatrix}0\\ \dot{\theta}_{sv}\\ 0\end{pmatrix}\right]^T\begin{bmatrix}\bar{i}_s\\ \bar{j}_s\\ \bar{k}_s\end{bmatrix} + \left[\mathbf{R}_1(\phi_{sv})\mathbf{R}_2(\theta_{sv})\begin{pmatrix}0\\ 0\\ \dot{\psi}_{sv}\end{pmatrix}\right]^T\begin{bmatrix}\bar{i}_s\\ \bar{j}_s\\ \bar{k}_s\end{bmatrix}$$

Equating the expressions for the components of the vector $\bar{\omega}_{sv}$ and manipulating we get

$$\dot{\phi}_{sv} = \Delta\omega_{xsv} + \dot{\psi}_{sv}\sin\theta_{sv}$$
$$\dot{\theta}_{sv} = \Delta\omega_{ysv}\cos\phi_{sv} - \Delta\omega_{zsv}\sin\phi_{sv}$$
$$\dot{\psi}_{sv} = (\Delta\omega_{ysv}\sin\phi_{sv} + \Delta\omega_{zsv}\cos\phi_{sv})/\cos\theta_{sv}$$

Integrating these gives the orientation of the seeker reference frame with respect to the variable window frame in terms of Euler angle rotations.

The initial conditions for the above angles are obtained from the initial transformation matrix given by

$$\mathbf{T}(\psi,\theta,\phi)_{sv} = \mathbf{T}(\psi,\theta,\phi)_{sE}\,\mathbf{T}^T(\psi,\theta,\phi)_{vE}$$

Where $(\psi,\theta,\phi)_{sE}$ are the initial input values of the seeker reference frame and $(\psi,\theta)_{vE}$ for example may be set equal to the initial attitude in the earth frame of the LOS to the primary target and ϕ_{vE} is an input parameter.

Then

$$\theta_{sv0} = \sin^{-1}\left(-\mathbf{T}_{sv}(1,3)\right)$$
$$\psi_{sv0} = \cos^{-1}\left(\mathbf{T}_{sv}(1,1)/\cos(\theta_{sv0})\right) * sign(\mathbf{T}_{sv}(1,2))$$
$$\phi_{sv0} = \cos^{-1}\left(\mathbf{T}_{sv}(3,3)/\cos(\theta_{sv0})\right) * sign(\mathbf{T}_{sv}(2,3))$$

Since the variable window frame can be oriented with respect to the earth frame to put the targets of interest near its XOY plane it only needs to be aligned with the HIL reference frame to make best use of a predominantly horizontal RF horn array. So in this case the motion table demands are $(\dot{\psi},\dot{\theta},\dot{\phi})_{sv}$ and $(\psi,\theta,\phi)_{sv}$ assuming the table rotation sequence is the same as that used in the model ie 321 the adopted convention. The orientation should be able to be tailored to suit any motion table/array set-up.

A.3.2 Motion table demands

The variable window frame is the linking frame and must be made coincident with the HIL reference frame. However further rotations may be required between the HIL reference and the table payload reference frame to accommodate different configurations of the motion table. Its attitude is fixed relative to the HIL reference so that it rotates at the same rate as the variable window frame in software space. Thus the demanded table rates can be calculated directly using the method given above.

Consideration should also be given to any misalignment of the missile hardware in the payload support rig. This could cause an unwanted resolution of the guidance command to a software autopilot and hence a distortion of the engagement instantaneous geometry. Large misalignments can cause a guidance failure.

Note the motion table rotation demands contain the variable window rotation so this places limits on the rotation rates of the variable window frame. If the variable frame is tied to the LOS frame then since it is possible to have very high LOS rates at intercept then some form of related limiting of table demands should be used. This is not a trivial problem.

A.4. Outputs to the RF Horn Array

A.4.1 Angle of arrival relative to the variable window frame

The angle of arrival information for each target is the attitude in the variable window frame of the LOS to that target from the missile. The simulation calculates the LOS Euler rates in the earth frame and these can be used to get the components $(\omega_x,\omega_y,\omega_z)_{iLL}$ in the ith LOS frame of the LOS rate $\bar{\omega}_{iL}$.

Thus for the ith target

$$\omega_{xiLL} = -\dot{\psi}_{iLE}\sin\theta_{iLE}$$
$$\omega_{yiLL} = \dot{\theta}_{iLE}$$
$$\omega_{ziLL} = \dot{\psi}_{iLE}\cos\theta_{iLE}$$

The rates $\bar{\omega}_v$, $\bar{\omega}_{iL}$, $\bar{\omega}_{iLv}$, $(\dot{\psi},\dot{\theta},\dot{\phi})_{iLv}$ and angles $(\psi,\theta,\phi)_{iLv}$ and initial angles of each LOS relative to the variable window frame are then calculated as above.

It was assumed above that the HIL reference and the scene generator frames were aligned. So, because of the variable window link, the variable window bore-sight would always point at the centre of the scene generator and its movement between targets would be embodied in the angle of arrival information ie $(\psi,\theta,\phi)_{iLv}$, calculated above.

The angle of arrival information is available for each radar target but a knowledge of which target the seeker is tracking may be significant as input to the algorithm which determines the variable window frame reference direction. Some missiles can process information on more than one target and will switch to the "stronger" target as the situation arises. If the variable frame is switched to the LOS from the missile to the tracked target then this behaviour will need to be accommodated by the table and the scene generator and possibly the tracker. This may be a step demand so it is probably best to have some filtering of the variable window frame rate that can soften step demands.

Information on which target is being tracked may not be available or readily calculated from simulation outputs. A possible scheme to solve this would be to align the variable window with the LOS to a centroid, of the targets, which is weighted by their radar cross-sections.

A.5. Outputs to the Missile

The use of variable reference frames in HIL simulations of missile engagements is inextricable linked with the injection of artificial signals, or forcing functions, into the missile. This concept is described in reference 1 which discusses the use of LOS rate injection in a limited HIL facility to expand its scope. The paper mentions the use of a variable reference frame as an extension of this to handle multiple targets.

A.5.1 Forcing function injection into missile hardware

The seeker processes the measured bore-sight error to generate a head spatial rate command ($\overline{\omega}_H$) which will cause the head to track the target and hence move at approximately the LOS rate ($\overline{\omega}_D \approx \overline{\omega}_L$). If a forcing function $\overline{\omega}_F = \overline{\omega}_L$ is combined with $\overline{\omega}_H$, the seeker movement is eliminated but for the error angle. The paper reports that given certain conditions the correct error angle history is essentially preserved.

Missiles that use the PN guidance law require an estimate of the LOS rate ($\dot{\omega}_L$) which the guidance unit uses to generate autopilot commands. Nesline and Zarchan[6] describe three ways to obtain this estimate using RF seekers. The conventional method uses filtered BSE measurements (similarly for some IR seekers). The LOS rate reconstruction scheme uses the seeker gyro output added to the bore-sight error rate as the estimate. The LOS reconstruction scheme uses bore-sight error measurements and Kalman filtering to obtain the estimate.

If the second scheme is used a forcing function will need to be added to the input to the guidance unit since LOS rate information is lost because the seeker is forced to remain almost stationary. This logically, is not necessary in the other cases. However there had been no local experience in using this technique viz, the artificial generation of LOS rate for HIL simulations, so these ideas needed to be tested. As a first step a simulation model with a fairly accurate model of the seeker and guidance unit processing was used.

A.5.2 Forcing function details

In the following analysis the underlying assumption is that there is an ideal space stabilisation loop which compensates for any missile body rotation. This loop can then be ignored.

If a variable window frame is used then the forcing function is $\overline{\omega}_F = \overline{\omega}_v$. So the head spatial rate demand, $\overline{\omega}_H - \overline{\omega}_v$, at the input to the tracker is compared with the head rate, $\overline{\omega}_G$, as measured by a rate gyro and in the steady state they are equal. If the variable window frame is tied to the LOS to the tracked target then the forcing function is ($\overline{\omega}_F = \overline{\omega}_v = \overline{\omega}_L$) and, in the steady state, the seeker will move at approximately $\dot{\omega}_L - \overline{\omega}_L = 0$ which is the negative of the bore-sight error rate. If the variable frame rate has an added component ie $\overline{\omega}_v = \overline{\omega}_L + \overline{\omega}_N$ then the seeker will move at approximately $\dot{\omega}_L - (\overline{\omega}_L + \overline{\omega}_N) \approx -\overline{\omega}_N$.

Note that the motion table demand will be the missile rate relative to the variable frame ie $\overline{\omega}_S - (\overline{\omega}_L + \overline{\omega}_N)$ so, when comparing with the basic configuration, the tracker rate with respect to the body will be preserved ie approximately $\dot{\omega}_L - \overline{\omega}_S$. Similarly the scene generator demand will be $\overline{\omega}_L - (\overline{\omega}_L + \overline{\omega}_N)$ and hence the tracking rate relative to the LOS will be preserved, ie approximately $\dot{\omega}_L - \overline{\omega}_L = 0$.

As can be seen the variable frame rate is subtracted from the demands to each hardware system. If their responses are perfect and the variable frame is not aligned with the earth frame then no extra errors will be introduced compared with the base configuration. However there will be errors because apart from the fact the responses are not perfect, they are also different. The errors may be minimised if the variable frame rate signal is filtered so as to compensate for the response of each piece of hardware. The other aspect of this to consider is that the bandwidth of the variable frame rate signal should be within the bandwidth of the hardware with the worst frequency response, although compensation filtering could be applied for this as well.

The forcing function $\overline{\omega}_v$ needs to be resolved into detector frame components $\omega_{xVD}, \omega_{yVD}, \omega_{zVD}$.

So

$$\begin{pmatrix} \omega_{xVD} \\ \omega_{yVD} \\ \omega_{zVD} \end{pmatrix} = \mathrm{T}(\psi,\theta,\phi)_{DS} \begin{pmatrix} \omega_{xVS} \\ \omega_{yVS} \\ \omega_{zVS} \end{pmatrix}$$

Where $\omega_{xVS}, \omega_{yVS}, \omega_{zVS}$ are calculated in 3.1 and $(\psi,\theta,\phi)_{DS}$, $\phi_{DS} = 0$ are the seeker head angles that need to be obtained from the hardware. The forcing functions ω_{yVD} and ω_{zVD}, are combined with the head pitch rate and yaw rate commands respectively.

A.6. Zero Motion Compensation

It is convenient to extend the idea of software and hardware space to think of the actual seeker unit in hardware space having its' virtual twin as part of the

missile airframe in software space. In normal HIL simulations using a motion table the two are almost coincident. However if the seeker is in a static rig and a single stationary source is used then the two will not coincide. Hence the hardware detector head angles are not preserved and the guidance commands to the autopilot will be incorrect. Therefore these commands must be resolved back into the detector frame using the actual head angles and then the correct resolution applied. Alternatively the detector frame guidance commands should be accessed and the correct resolution applied using the same scheme as implemented in the missile. Note that whether a hardware autopilot or a software model autopilot is used they both "occupy" software space.

What is required is the attitude of the virtual seeker with respect to the hardware unit. Obtaining the transformation matrix between these frames most conveniently does this.

The transformation matrix from the variable window to the virtual seeker reference frame is

$$\mathbf{T}_{sv} = \mathbf{T}_{vE}\mathbf{T}^{I}{}_{vE}$$

And that from hardware unit to the virtual seeker is

$$\mathbf{T}_{su} = \mathbf{T}_{sv}\mathbf{T}^{T}{}_{uv}$$

Where $\mathbf{T}_{tv}$ is constant and is obtained from the attitude of the unit with respect to the variable window frames or HIL reference frame. This transformation may include a payload to HIL reference as well as a misalignment matrix. (All 3 assumed coincident in this case)

The transformation matrix from the unit detector to the virtual seeker reference frame is thus

$$\mathbf{T}_{DS} = \mathbf{T}_{DU}\mathbf{T}^{T}{}_{SU}$$

Where $\mathbf{T}_{DU}$ is obtained from the actual head angles. Elements of $\mathbf{T}_{DS}$ can then be used in the correct resolution of the detector frame guidance commands to the autopilot frame.

The forcing functions are obtained by resolving the components of the variable window frame rotation rate as follows

$$\begin{pmatrix} \omega_{xVD} \\ \omega_{yVD} \\ \omega_{zVD} \end{pmatrix} = \mathbf{T}(\psi,\theta,0)_{DU}\,\mathbf{T}(\psi,\theta,\phi)_{UV} \begin{pmatrix} \omega_{xVv} \\ \omega_{yVv} \\ \omega_{zVv} \end{pmatrix}$$

The target source should be on the X axis of the HIL reference frame. Note that any unaccounted for misalignments here or of the unit in the payload rig will change the bore-sight error history and hence distort the engagement geometry from the true geometry.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37252

AIAA-2000-4291

Investigation of Mathematical Models of Otolith Organs for Human Centered Motion Cueing Algorithms

Robert J. Telban*
Frank M. Cardullo†
Liwen Guo‡
State University of New York
Binghamton, New York

Abstract

The authors, as part of their work in human centered motion cueing algorithms, conducted research in the area of mathematical modeling of the otolith organs, the sensors of specific force. The purpose of this study was to develop a model that is consistent with both experimental and theoretical analyses that can be readily implemented into a motion cueing algorithm.

The authors reviewed several existing models that characterize the specific force response dynamics. Experimental research on the ocular torsion response of human subjects[8] resulted in a second-order model with a first-order lead component, with a short time constant of 0.66 seconds. From this model and physiological knowledge, a first-order lead-lag model of the otolith afferent dynamics was estimated[2]. A second-order lumped parameter model for the otolith displacement as a function of the specific force stimulus was derived[11], revealing a short time constant of 0.0002 seconds. Physiological experiments measuring the afferent response[6] resulted in an otolith mechanics time constant of 0.016 seconds. Based upon these results found in the literature, the authors synthesized a new otolith model.

The physiological experiments also resulted in models for both regular and irregular units containing a fractional exponent term in the Laplace domain transfer function. The authors derived time responses for these models using fractional calculus with a series approximation. The responses from these models are compared to the response obtained from the proposed model. This comparison illustrates that the proposed model is a reasonable approximation to the more complex physiological models.

Introduction

The otolith organs are located in the inner ear and provide linear motion sensation in humans and mammals. These organs are responsive to specific force, the gravitoinertial reaction force per unit mass, which is defined to be $\underline{f} = \underline{g} - \underline{a}$,[1,2] where $\underline{g}$ is the local gravitational force vector, and $\underline{a}$ is the acceleration of the head with respect to a body-fixed reference frame. Therefore, the otoliths respond to both linear acceleration and tilting of the head with respect to the gravity vector. However, the otoliths cannot discriminate between acceleration and tilt, requiring additional sensory information to resolve this ambiguity. This phenomenon can be used to advantage in motion simulation. There are two otolith organs, the utricle and saccule in each inner ear. The utricle primarily senses motion in the horizontal plane, while the saccule primarily senses motion in the vertical plane.

The orientation of the otolith organs with respect to a body-fixed reference frame located on the head is shown in Figure 1. The otolith reference frame is fixed to the head; thus motion in this frame is relative to the head. The x-z plane of the otolith reference frame is tilted upward from the x-axis by about 20 degrees.[3] The utricle is oriented along the x-axis and the saccule is oriented along the z-axis in the otolith reference frame.

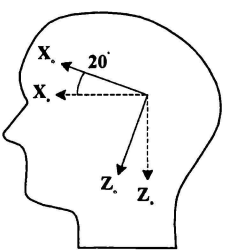


Figure 1. Orientation of the Otolith Organs and Body-Fixed Reference Frame.

---

* PhD Candidate, Department of Mechanical Engineering, Student Member AIAA
† Associate Professor, Department of Mechanical Engineering, Associate Fellow AIAA
‡ PhD Student, Department of Mechanical Engineering.

"Copyright © 2000 by the American Institute of Aeronautics and Astronautics. All rights Reserved."

Physiological Description of the Otoliths

The otolith organs consist of a two-layer structure known as the otolithic membrane that is attached to a base containing sensory cells. The otolithic membrane is composed of an upper layer, the otoconial layer, and a lower layer, the gelatinous layer. A fluid known as endolymph is in contact with the upper surface of the otoconial layer. The otoconial layer consists of calcium carbonate crystals embedded in a gelatinous material that rests on a less dense and extremely deformable gelatinous layer. This gelatinous layer is in turn attached to the sensory cell base known as the macula that is incorporated into the membranous tissue walls of the inner ear. The macula is rigidly attached to the skull and therefore moves with the head.

There are two types of sensory cells located in the macula. The Type I cells are enclosed in a nerve chalice and are innervated by nerve fibers with a large diameter. The Type II cells are cylindrical and are innervated by fibers with a small diameter. Fernandez and Goldberg[4] report that cells in the outer (peripheral) otolith region are primarily Type II cells and cells in the central (striolar) region are primarily Type I cells. Both types of cells have a series of small hairs that penetrate the lower portion of the gelatinous layer. Each hair cell has about 70 stereocilia and one kinocilium with the stereocilia graded in length toward the kinocilium.

The resulting displacement of the otolithic membrane due to forward linear acceleration is illustrated in Figure 2. The arrows in the figure show the direction of the specific force acting upon the head. With a forward acceleration or backward tilting of the head the denser otoliths tend to lag behind the macula, with the relative motion resulting in deforming the gelatinous and otoconial layers in shear. When the shear deformation is in the direction of the kinocilium, the cell will be excited, whereas when the deformation is in the opposite direction, the cell will be inhibited. The directions of the maximum excitation and inhibition of a hair cell are defined by its polarization axis. In each macula, the central parting known as the striola separates oppositely polarized regions. For each position due to translational movement, some cells will be maximally excited, while others will be maximally inhibited.

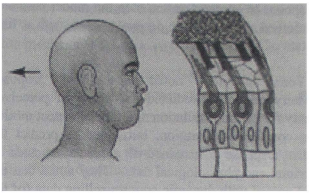


Figure 2. Displacement of the Otolithic Membrane due to a Forward Acceleration.[5]

The axes of maximum and minimum response of a given afferent neuron are defined by the corresponding polarization axis of the hair cells that it innervates. The linear polarization of an afferent neuron strongly suggests that the hair cells that the neuron innervates have polarization axes that are oriented in the same direction.[3,4] The response of a neuron is the afferent firing rate (AFR), measured in impulses per second (ips).

Fernandez and Goldberg[6] identified two types of neurons that are characterized by their variance or regularity of discharge, hereafter referred to as regular and irregular units. From a sample population of units, they identified a ratio of regular to irregular units to be approximately three to one.

Empirically Based Models of Perceived Motion

Zacharias[7] reported that Meiry first investigated subjective responses to linear motion by using a cart to produce longitudinal sinusoidal motion. By measuring the subjective indication of direction, he obtained a transfer function relating perceived velocity $\hat{v}$ to actual velocity v:

$$\frac{\hat{v}(s)}{v(s)} = \frac{K\tau_1 s}{(\tau_1 s + 1)(\tau_2 s + 1)} \qquad (1)$$

Where the long time constant τ_1 and short time constant τ_2 are 10 and 0.66 seconds respectively, and the gain K is undetermined since amplitude measurements were not taken. Zacharias[7] then noted that Peters suggested

the subjective response measured by Meiry was perceived acceleration and not perceived velocity, since in response to an acceleration step the model predicted a perceived response that decays to zero with a time constant of 10 seconds.

Young and Meiry[8] noted that the model proposed by Meiry correctly predicted the phase of perceived velocity for lateral oscillation and time to detect motion under constant acceleration, but failed to predict the otoliths' response to sustained tilt angle as indicated by behavioral and physiological data. They noted that the model agreed with dynamic counter-rolling data (of the eye) at high frequencies, but experimental counter-rolling at zero frequency showed a static component of otolith output with no phase lag (The model assumed no static output and at zero frequency approached 90 degrees of lead). They proposed the following revised model of specific force sensation:

$$\frac{\hat{f}(s)}{f(s)} = \frac{1.5\,(s + 0.076)}{(s + 0.19)(s + 1.5)} \qquad (2)$$

which, when rearranged in terms of the time constants, yields

$$\frac{\hat{f}(s)}{f(s)} = \frac{0.4\,(13.2s + 1)}{(5.33s + 1)(0.66s + 1)} \qquad (3)$$

With a smaller long time constant (5.33 seconds) and an additional lead term, they modeled both perceived tilt and acceleration in response to acceleration input. They noted that the model acts as a velocity transducer over the frequency range of 0.19 to 1.5 rad/s, with the transfer function from specific force to perceived tilt or lateral acceleration having a static sensitivity of 0.4.

Analytical Model of Otolith Dynamics

Zacharias[7] noted that a mass-spring-dashpot model of otolith motion could be used to represent the two lag time constants, similar to the torsion-pendulum model for the semicircular canals. Ormsby[2] first developed this model, and Grant, Best, et al[9,10,11,12] later refined the model as part of their theoretical analysis of the otolithic membrane. A lumped parameter model is constructed by considering the forces on the otoconial layer. The differential equation for this model in the x-direction (in the otolith reference frame) is given as[11]

$$-b\dot{x} - kx + m_o g_x + m_{df}\,(a_x - g_x) = m_o a_B \qquad (4)$$

where

m_o = Mass of the displaced otoconial layer
m_{df} = Mass of the displaced fluid (endolymph)
g_x = Component of the gravity vector
b = Viscous damping on the otoconial layer
k = Stiffness of the gelatinous layer
x = Displacement of the otoconial layer with respect to the head
a_x = Acceleration of the head with respect to a body-fixed reference frame
a_B = Acceleration of the otoconial membrane with respect to a body frame

The force term of $m_{df}(a_x - g_x)$ is the buoyant force of the endolymph acting on the otoconial layer, which was neglected by Ormsby[2] in his analysis.

By conservation of volume, the mass of the displaced fluid can be expressed as $m_{df} = (\rho_e / \rho_o)\, m_o$, where ρ_e is the density of the endolymph and ρ_o is the density of the otoconial membrane. Substituting this term into Eqn. (4), and noting that $a_B = a_x + \ddot{x}$, results in

$$\ddot{x} + \frac{b}{m_o}\dot{x} + \frac{k}{m_o}x = \left(1 - \frac{\rho_e}{\rho_o}\right)(g_x - a_x) \qquad (5)$$

Note that the stimulus term $g_x - a_x$ is the specific force component f_x. The term $(1 - \rho_e / \rho_o)$ establishes the system sensitivity or gain to the stimulus terms. For ρ_o greater than ρ_e, a positive g_x or a negative a_x will produce a positive displacement of the otoconial layer. Eqn. (5) can then be written in transfer function form as

$$\frac{x(s)}{f(s)} = \left(1 - \frac{\rho_e}{\rho_o}\right)\frac{1}{\left(s^2 + \frac{b}{m_o}s + \frac{k}{m_o}\right)} \qquad (6)$$

As observed by Young and Meiry[8], the system response is overdamped. For an overdamped system, Eqn. (6) can be rewritten in terms of the long and short time constants as

$$\frac{x(s)}{f(s)} = \left(1 - \frac{\rho_e}{\rho_o}\right)\frac{\tau_1\tau_2}{(1+\tau_1 s)(1+\tau_2 s)} \qquad (7)$$

where for the otoliths. $\tau_1 \gg \tau_2$, and therefore the two time constants can be related to the lumped parameters as $\tau_1 = m_o/k$ and $\tau_2 = m_o/b$.

In determining the value of the short time constant τ_2, Grant and Best[11] examined the maximum displacement of the otoconial layer in response to a step change in linear velocity. The acceleration for a linear velocity step U is $a_x = -U\delta(t)$, with $g_x = 0$, where $\delta(t)$ is the unit impulse function. The transient response to Eqn. (7) is then

$$x(t) = U\left(1 - \frac{\rho_o}{\rho_e}\right)\tau_2\left(e^{-t/\tau_1} - e^{-t/\tau_2}\right) \qquad (8)$$

By assuming that the short exponential term in Eqn. (8) has reached zero and the long exponential term remains close to unity, the maximum displacement of the otoconial layer x_{max} can be approximated as

$$x_{max} \cong U\left(1 - \frac{\rho_e}{\rho_o}\right)\tau_2 \qquad (9)$$

The theoretical continuum mechanics analysis performed by Grant and Best first indicated that this short time constant τ_2 is 0.002 seconds or less[10]. They later demonstrate[11] that this value turns out to be too large when reasonable values of the maximum otolith displacement are considered. For $\rho_o = 2.0$ and $U = 25$ cm/sec (a reasonable value for normal head velocity), Eqn. (9) becomes $x_{max} = 12.5\tau_2$. For $\tau_2 = 0.002$ sec, the maximum displacement of the otolithic membrane results in $x_{max} = 250$ μm. It is assumed that for shear deformation the maximum displacement should not exceed the thickness of the otoconial layer (25 μm), indicating the short time constant should be one order of magnitude smaller, i.e. $\tau_2 = 0.0002$ s. This indicates that more damping is needed in the lumped parameter model. Grant and Best later show that additional damping can be introduced by inclusion of a viscoelastic gelatinous layer in the continuum mechanics model[12].

Estimated Model of Afferent Dynamics

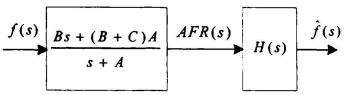
Ormsby[2] neglected the short time constant τ_2 in Eqn. (7) and after rearranging terms, approximated the otolith mechanical dynamics by

$$\frac{x(s)}{f(s)} = \frac{A}{s+A} \qquad (10)$$

and then proposed a model for the response of the otolith afferent dynamics:

$$\frac{AFR(s)}{f(s)} = \frac{Bs + (B+C)A}{s+A} \qquad (11)$$

This model assumes that higher centers process the afferent response optimally to estimate the perceived specific force $\hat{f}$ as shown below.



Combined Mechanical and Afferent Otolith Dynamics

Processing by Higher Centers

The steady-state optimal processor $H(s)$ is then determined by solving the associated Wiener-Hopf equation, yielding a solution of the form

$$H(s) = M\frac{s+A}{(s+F)(s+G)} \qquad (12)$$

where F, G, and K are non-linear functions of a set of independent variables that include the parameters A, B, and C in Eqn. (11). With the form of $H(s)$ determined, it can then be cascaded with the otolith and afferent dynamics to estimate the perceptual response:

$$\frac{\hat{f}}{f} = BM\frac{(s+(B+C)A/B)}{(s+F)(s+G)} \qquad (13)$$

which is equivalent to Eqn. (2).

Ormsby[2] noted that Fernandez, Goldberg, and Abend found an average steady-state change in afferent firing rate from the utricle due to a 1 g step to be 45 impulses per second (ips), resulting in the condition that $B + C = 45$. Setting Eqn. (13) equal to Eqn. (2) and including this constraint results in the following model for the afferent dynamic response:

$$\frac{AFR(s)}{f(s)} = 90\,\frac{s+0.1}{s+0.2} \tag{14}$$

This transfer function, when rearranged in terms of its time constants, becomes

$$\frac{AFR(s)}{f(s)} = 45\,\frac{10s+1}{5\cdot+1} \tag{15}$$

Ormsby[2] noted the following about the model:

"The approach taken here can yield a model which accounts reasonably well for the available subjective data, the known physiological structure of the sensor and makes reasonable predictions concerning the afferent processes and the associated central processing."

Experimental Models of Afferent Dynamics

Fernandez and Goldberg[4] studied the discharge of peripheral otolith neurons in response to sinusoidal force variations in the squirrel monkey. Both regular and irregular units were measured, with a frequency analysis performed for each type of unit. The gain curves for the regular units were relatively flat, with a small phase lead at low frequencies and a larger phase lag at high frequencies. The irregular units showed a larger gain enhancement and phase lead at high frequencies. On average, there is an increase by a factor of 18 in gain enhancement in irregular units but only an increase of a factor of 2 for regular units. In both cases, a first-order lead operator cannot represent the resulting gain enhancement and phase lead. The average static sensitivity for both types of units is nearly identical, with reduced gains for the inhibitory response.

The frequency responses of regular and irregular units resulted in a transfer function of the form[4]

$$\frac{AFR(s)}{f(s)} = G_S\,\frac{1+k_A\tau_A s}{1+\tau_A s}\,\frac{1+k_v(\tau_v s)^{k_v}}{1+\tau_M s} = G_S\,H_A(s)\,\frac{H_v(s)}{H_M(s)} \tag{16}$$

In Eqn. (16), the term H_v is a velocity-sensitive operator with a fractional exponent ($k_v < 1$) and provides most of the gain enhancement and phase lead found in both irregular and regular units. The value of k_v reflects the effectiveness of the lead operator and is closely related to the slope of the gain curve. The term H_A is an adaptation operator that contributes to low frequency phase leads and gain increases from static or zero frequency to 0.006 Hz. The term H_M is a first-order lag operator that Fernandez and Goldberg[4] note may reflect the mechanics of otolith motion. This lag term accounts for the high frequency phase lags observed in regular units and for high frequency phase leads in irregular units being smaller than would be predicted solely by a fractional lead operator. The term G_S defines the static sensitivity in terms of afferent firing rate per unit of acceleration, i.e. ips/g.

The transfer function was estimated from a least square computer fit with τ_v varied from 0 to 320 seconds in seven steps, with the remaining parameters estimated. The values for these parameters were obtained for $\tau_v = 40$ seconds (almost equal results were obtained for all values of τ_v). The median parameters for both regular and irregular units for the excitatory response are given in Table 1.

Table 1. Median Parameters for Regular and Irregular Units[4].

Parameter	Regular Unit	Irregular Unit
k_v	0.188	0.440
k_A	1.12	1.90
τ_A	69 sec	101 sec
τ_M	16 msec	9 msec
G_S	25.6 ips / g	20.5 ips / g

Proposed Afferent Dynamics Model

Note that the gain terms for the Fernandez-Goldberg model from Table 1 are about one half that of the gain value used by Ormsby[2] to develop his model. Due to the adaptation mechanism in the Fernandez-

Goldberg models, these gains will require a long duration step input to be realized in steady state. Hosman[13] suggested a gain term of less magnitude that that used by Ormsby (G_S = 33.3) that may provide an improved approximation to the Fernandez-Goldberg responses.

By using the long and lead time constants reported by Ormsby[2] in Eqn. (15), selecting the short time constant from the Fernandez-Goldberg[4] model, and including the gain suggested by Hosman, the following transfer function results for the afferent otolith dynamics:

$$\frac{AFR(s)}{f(s)} = 33.3\frac{(10s+1)}{(5s+1)(0.016s+1)} \qquad (17)$$

The frequency response of the proposed model in Eqn. (17) is compared to the frequency response of the Young-Meiry model of Eqn. (3) as shown in Figure 4. For comparison in Figure 4, both models use the gain K = 0.4 from the Young-Meiry model. Note that the gain and phase lag for the Young-Meiry model occurs at a much lower frequency as compared to the proposed model. This is due to the magnitude of the short time constant τ_2 for the Young-Meiry model being an order of magnitude larger than the value used in the proposed model. In the range of normal head movements from 0.1 to 1.0 Hz noted by Young[1], the gain for the proposed model remains constant with the phase close to zero degrees. In this frequency range the otolith functions as a specific force transducer.

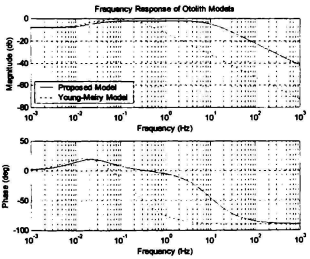


Figure 4. Frequency Response of Proposed and Young-Meiry Sensation Models.

Fractional Exponent Derivation

Because of the fractional exponent in the transfer function of Eqn. (16), an elementary solution to its response cannot be readily obtained. However, an approximate solution to the response can be derived through the application of fractional calculus[14].

By first substituting the regular unit parameters into Eqn. (16) and then implementing partial fraction expansion, Eqn. (16) becomes

$$H(s) = \frac{1792.056}{s+62.5} + 674.058\frac{s^{0.188}}{s+62.5} - \frac{0.044538}{s+0.0145} - 0.016752\frac{s^{0.188}}{s+0.0145} \qquad (18)$$

In Eqn. (18) there are two groups of two transfer functions. Each group is related to either the otolith mechanics ("fast") time constant τ_M or the adaptation ("slow") time constant τ_a, with one of the two transfer functions including an exponent that represents a fractional derivative. For the first group, the solution to the term without the fractional exponent can be easily obtained by taking the inverse Laplace transformation of the response:

$$L^{-1}\left(\frac{1}{s+62.5}\right) = e^{-62.5t} \qquad (19)$$

To derive a solution to the fractional exponent term, The inverse Laplace transformation is first obtained by applying fractional calculus:[14]

$$L^{-1}\left(\frac{s^{-\nu}}{s-a}\right) = E_t(\nu, a) \qquad (20)$$

where a = -62.5, ν = -0.188, and the term $E_t(\nu,a) = t^{\nu}e^{at}\gamma^*(\nu,at)$, with γ^* being the incomplete gamma function[14], a transcendental function that can be expressed as

$$\gamma^*(\nu,at) = t^{-at}\sum_{k=0}^{\infty}\frac{(at)^k}{\Gamma(\nu+k+1)} \qquad (21)$$

Substituting Eqn. (21) into Eqn. (20) will result in

$$\mathcal{L}^{-1}\left(\frac{s^{-\nu}}{s-a}\right) = E_t(\nu, a) = t^{\nu}\sum_{k=0}^{\infty}\frac{(at)^k}{\Gamma(\nu+k+1)} \tag{22}$$

Eqn. (22) is an infinite series. For $\nu = 0$, Eqn. (22) will reduce to the Taylor series expansion of the exponential function:

$$E_t(0, a) = \sum_{k=0}^{\infty}\frac{(at)^k}{\Gamma(k+1)} = e^{at} \tag{23}$$

When ν is not equal to zero, $E_t(\nu, a)$ is a transcendental function that cannot be expressed by an elementary function, i.e. it can only be approximated. However, Eqn. (22) is not suitable for numeric computation because this series converges very slowly, especially when the magnitude of a is large (e.g. -62.5). The following derivation will result in an integral expression of the infinite series that can be adopted to compute $E_t(\nu, a)$. If we let $f(t) = E_t(\nu, a)$, then

$$f'(t) = t^{\nu-1}\sum_{k=0}^{\infty}\frac{(at)^k}{\Gamma(\nu+k)} \tag{24}$$

where $f(t)$ satisfies the first-order ordinary differential equation

$$f'(t) - af(t) = t^{\nu-1}/\Gamma(\nu) \tag{25}$$

with the solution

$$f(t) = E_t(\nu, a) = \frac{e^{at}\int_0^t e^{-au}u^{\nu-1}du}{\Gamma(\nu)}, \nu \geq 1 \tag{26}$$

Note that in Eqn. (26) the integral does not exist when ν - 1 is less than 0. To overcome this problem, we use the recursion formula[14]

$$\begin{aligned} E_t(\nu, a) &= \frac{t^{\nu}}{\Gamma(\nu+1)} + aE_t(\nu+1, a) \\ &= \frac{t^{\nu}}{\Gamma(\nu+1)} + \frac{at^{\nu+1}}{\Gamma(\nu+2)} + \frac{a^2e^{at}}{\Gamma(\nu+2)}E_t(\nu+2, a) \\ &= \frac{t^{\nu}}{\Gamma(\nu+1)} + \frac{at^{\nu+1}}{\Gamma(\nu+2)} + \frac{a^2e^{at}}{\Gamma(\nu+2)}\int_0^t e^{-au}u^{\nu+1}du \end{aligned} \tag{27}$$

Note that the integral in Eqn. (27) now exists since $\nu + 1$ is greater than 0. Eqn. (27) can now be used to compute the responses of the two fractional exponent transfer functions given in Eqn. (18). For each of these transfer functions, three terms are computed. The first two terms are analytical functions, and the third includes an integral that requires an approximate solution. By using a series approximation, the integral *I(t)* in Eqn. (27) can be evaluated as

$$I(t) = t^{\nu+2}e^{-at}\left\{\sum_{j=0}^{N}C_j\int_0^t e^{at(1-z)}(z-1)^j\,dz + O\left(\frac{1}{(-at)^{N+1}}\right)\right\} \tag{28}$$

where the constant C_j is equal to 1 for $j = 0$, $(\nu + 1)$ for $j = 1$, and $(\nu + 1)\nu/2!$ for $j = 2$. By taking the inverse Laplace transformation of Eqn. (18) and applying Eqns. (27) and (28) to the transfer functions with fractional exponents results in the impulse response *h(t)*:

$$\begin{aligned} h(t) = {} & 1792.056e^{-62.5t} - 0.044538e^{-0.0145t} \\ & + 674.058\,E_t(-0.188, -62.5) \\ & - 0.016752\,E_t(-0.188, -0.0145) \end{aligned} \tag{29}$$

The response to a step input will now be considered. Given a system with the initial conditions $x = 0$ and $\dot{x} = 0$ when $t = 0$, and an arbitrary input *u(t)*, we look for a solution in the form

$$x(t) = \int_0^t h(x, \tau)u(\tau)d\tau \tag{30}$$

where $h(x, \tau)$ is Green's function, i.e. the system response to an impulse input. If we consider the response to a unit step, i.e. $u(t) = 1$ for t > 0, the response for a term without the fractional exponent is

$$\int_0 e^{a\tau} d\tau = \frac{1}{a}\left(e^{at} - 1\right) \tag{31}$$

while the response for a term with the fractional exponent from Eqn. (29) is given by Miller and Ross[14] as

$$\int_0 E_\tau(v, a)\, d\tau = E_t(v+1, a) \tag{32}$$

and applying the recursion formula in Eqn. (27) gives

$$E_t(v+1, a) = \frac{1}{a}\left(E_t(v, a) - \frac{t^v}{\Gamma(v+1)}\right) \tag{33}$$

Applying Eqns. (31) and (33) to the impulse response given in Eqn. (29) and combining terms results in the regular unit response to a unit step:

$$\begin{aligned} x(t) = {} & 25.601 - 28.673e^{-62.5t} + 3.073e^{-0.014493t} \\ & - 10.786E_t(-0.188, -62.5) \\ & + 1.156E_t(-0.188, -0.014493) \\ & + 9.629\frac{t^v}{\Gamma(v+1)} \end{aligned} \tag{34}$$

Similarly, the unit step response for the irregular unit can be derived:

$$\begin{aligned} x(t) = {} & 20.308 - 35.588e^{-111.1111t} + 18.280e^{-0.009901t} \\ & - 86.063E_t(-0.44, -111.1111) \\ & + 40.769E_t(-0.44, -0.009901) \\ & + 45.294\frac{t^v}{\Gamma(v+1)} \end{aligned} \tag{35}$$

Comparisons of Model Responses

The response to a unit step input of 1 g (9.81 m/s^2) will now be examined for both the regular and irregular units. A series approximation of $N = 2$ will be used, with the responses evaluated at intervals of 0.025 seconds. The effect of the two fractional exponent lead terms from the otolith mechanics ("fast") and adaptation ("slow") time constants will be illustrated. Figure 5 shows the response for the regular units for 1 second. Note that the rise time is faster and the steady state response significantly increases due to the fast lead term, with a smaller but additional effect due to the slow lead term.

Figure 6 shows the response for the irregular units for 1 second. Note that both lead terms have a more significant effect on the response as compared to their effect on regular unit response. This is primarily due to the larger fractional exponent k_v, and also due to the larger time constant τ_a and coefficient k_a in the adaptation operator. The rise time is significantly faster, with the response rapidly increasing to a peak overshoot over five times the value without the fast lead term. The inclusion of the fast lead term also results in the steady state response being nearly doubled. The addition of the slow lead term results in the overshoot increasing by an additional 50 per cent with additional steady state response.

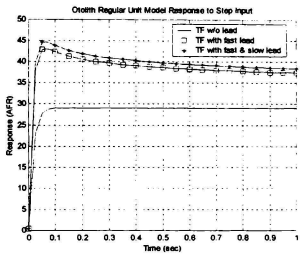


Figure 5. Regular Unit Response to a Unit Step Input.

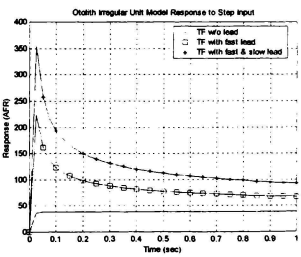


Figure 6. Irregular Unit Response to a Unit Step Input.

Figure 7 compares the step response for both the regular and irregular units (including both the fast and slow lead terms) to the response for the proposed model given in Eqn. (17). Note that the rise time for the proposed model is faster than the regular unit, but slower than the irregular unit. There is no large overshoot as observed with the irregular unit response. Figure 8 compares the responses for 30 seconds. Note that the steady state response for the proposed model is less than the irregular unit response but greater than the regular unit response, and approaches the regular unit response for the given time duration. Both the regular and irregular unit response will slowly approach their respective gain values, and beyond about 80 seconds the irregular unit response will decrease below that of the proposed model.

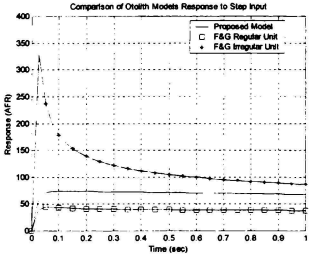


Figure 7. Proposed Model Response to a Unit Step Input for 1 second.

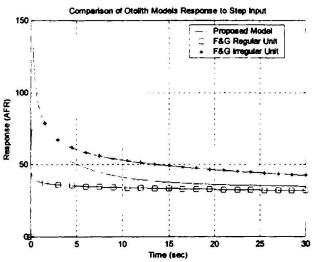


Figure 8. Proposed Model Response to a Unit Step Input for 30 seconds.

Physiological Interpretation

Modern theories of the operation of the otolith receptors are based on the assumption that the neural impulses are generated by the deflection of hairs in the sensory cells as a result of the otolith displacement. Specific force, produced by either linear acceleration or tilt, is first transformed into electrical impulses by the otolith-endolymph system. Then the otolith deflection is further transformed into electrical impulses by the mechano-neural transduction system consisting of sensory hair cells, afferent nerves, and efferent nerves.

Many researchers have shown that the otolith-endolymph system could be represented by an overdamped mass-spring-damper system. Grant and Best[11] report that the magnitude of the long time constant τ_1 is considered correct by most investigators because the overall system (otolith organ, nervous transmission, central nervous system processing, and eye motion dynamics) could easily follow such a slow system. The value Grant and Best[11] obtain for the short time constant τ_2 is a three order of magnitude decrease in time constant as compared to the value obtained from the ocular torsion responses measured by Young and Meiry[8]. This value is also two orders of magnitude less than the value of τ_M Fernandez and Goldberg[4] attribute to the otolith dynamics. The dynamic response increases as the system transducer (otolith) is approached, thus allowing for dynamic losses in nervous system transmission and eye dynamics.

Young and Meiry[8] first noted that the origin of the lead term could be neurological, either in central processing of the otolith displacement signals or through the presence of two types of hair cells in the macula. One type of hair cell would respond to displacement and the other would respond to the rate of change of otolith displacement. These hair cells could produce the lead term if they were of the slowly adapting type postulated by several researchers. Fernandez and Goldberg[4] later show that the degree of sensitivity to the otolith velocity can be represented by different fractional exponents in the lead operator, i.e. irregular units are more velocity sensitive than regular units. They note that this difference in sensitivity may be due to discrepancies that are more noticeable with irregular units.

Fernandez and Goldberg[4] suggest that the difference between the expected otolith displacement

and afferent firing rate for both regular and irregular units may be attributed to the mechanical linkages between the sensory hair bundles and the gelatinous layer. They report that these sensory hair bundles are not rigidly embedded in the membrane, but are enclosed in a fluid-filled meshwork between the membrane and the sensory epithelium. Motion could be transferred to the hairs either by directly contacting the meshwork or indirect contact by viscous coupling with the fluid. They also note that the irregular units correspond to thick afferents that stimulate the Type I hair cells in the striola. Grant and Best[11] also suggest that for large tilt the gelatinous layer has a nonlinear stiffness that could also contribute to these differences as well.

Fernandez and Goldberg[4] suggest that later stages of the transduction process may also influence the afferent dynamics, in particular adaptation, noting the following:

"Lowenstein found, in the ray, that afferents can show adaptation when galvanic currents are applied. We have made similar observations in the squirrel monkey (unpublished observations) and have noted a correlation between the degree of adaptation shown by a particular afferent in its response to natural and electrical stimulation"

They note that these results show that a detailed comparison of the response to galvanic and natural stimulation may help in assessing the contributions made by the various transduction stages in the filtering process between the motion of the otolith and the discharge of the afferent.

Conclusions

A mathematical model of the otolith organs based on both analytical modeling and physiological experiments is proposed. This model consists of a second-order mass-spring-dashpot operator cascaded with a first-order lead operator. The mass-spring-dashpot operator represents the mechanics of otolith motion. The lead operator arises from the mechano-neural transduction system that in turn generates the afferent response.

The physiological experiments resulted in transfer functions for both regular and irregular units with a fractional exponent in the lead operator. This term reflects the sensitivity of hair cells to the rate of change of otolith displacement. By applying fractional calculus, transient responses to impulse and step inputs have been derived for both regular and irregular unit models. The solutions to these responses are both mathematically and computationally intensive, requiring several terms along with a series approximation of an integral for each of two lead components to compute the response at each time step.

Additional research in this area is needed to yield responses to more arbitrary inputs in addition to the impulse and step inputs. It is also suggested that a model could be developed that incorporates the dynamics of the fractional exponent lead operator from both the regular and irregular unit models. Such a model must be easy to implement in state space in a motion cueing algorithm and be computationally efficient so that the afferent response can be computed in real time.

Comparison of the transient response of the proposed model with the responses of the regular and irregular units clearly shows that a less complex model can generate a response that is a reasonable approximation between the regular and irregular units. This model has the same structure as the Meiry-Young model that is currently used in the development of the optimal algorithm[15], and can easily replace the former model. This revised otolith model will also be an integral part of a proposed motion cueing algorithm that incorporates human perception of motion into a non-linear optimal control structure.

References

[1] Young, L. H., Perception of the Body in Space: Mechanisms, Handbook of Physiology – The Nervous System III, Chapter 22, pp. 1023 – 1066.

[2] Ormsby, C. C., Model of Human Dynamic Orientation, Thesis, Massachusetts Institute of Technology, Cambridge, MA, January 1974.

[3] Howard, I. P., Chapter 11, The Vestibular System, pp. 11-1 – 11-30, in Handbook of Perception and Human Performance, Volume I, Sensory Processes and Perception, Boff, K. R., Kaufman, L., and Thomas, J. P, ed., John Wiley and Sons, Inc., New York , 1986.

[4] Fernandez, C. and Goldberg, J. M., Physiology of Peripheral Neurons Innervating Otolith Organs of the Squirrel Units, III: Response Dynamics. Journal of

Neurophysiology, v. 39, no. 5, September 1976, pp. 996 – 1008.
[5] Purves, Augustine, Fitzpatrick, Katz, LaMantia, & McNamara, Neuroscience, Sinauer Associates, Inc, Sunderland, MA, 1997.
[6] Fernandez , C. and Goldberg, J. M., Physiology of Peripheral Neurons Innervating Otolith Organs of the Squirrel Monkey, I: Response to Static Tilts and to Long-Duration Centrifugal Force., Journal of Neurophysiology, v. 39, no. 5, September 1976, pp. 970 – 983.
[7] Zacharias, G., Motion Cue Models for Pilot-Vehicle Analysis, Defense Documentation Center, Defense Logistics Agency, AMRL-TR-78-2, May 1978.
[8] Young, L. R., and Meiry, J. L., A Revised Dynamic Otolith Model, Aerospace Medicine, June, 1968, vol. 39, no. 6, pp. 606 – 608.
[9] Grant, J. W., Best, W.A., and LoNigro, R., Governing Equations of Motion for the Otolith Organs and their Response to a Step Change in Velocity of the Skull, Journal of Biomechanical Engineering, Transactions of the ASME, vol. 106, no. 4, Nov. 1984, pp. 302 – 308.
[10] Grant, J. W., Best, W. A., Mechanics of the Otolith Organ – Dynamic Response, Annals of Biomedical Engineering, vol. 14, 1986, pp. 241 – 256.
[11] Grant, W. and Best, W., Otolith-Organ Mechanics: Lumped Parameter Model and Dynamic Response, Aviation, Space, and Environmental Medicine, October, 1987, pp. 970 – 976.
[12] Grant, W. and Best, W., Viscoelastic Effects in Distributed Parameter Otolith Modeling, ASME Applied Mechanics Division Symposia Series, v. 84, 1987, pp. 299 – 302.
[13] Hosman, R. J., Pilot's Perception and Control of Aircraft Motions, Thesis, Delft University of Technology, The Netherlands, 1996.
[14] Miller and Ross, An Introduction to the Fractional Calculus and Fractional Differential Equations, John Wiley and Sons, Inc., New York, 1993.
[15] Telban, R. J., Cardullo, F. M., and Houck, J. A., Developments in Human Centered Cueing Algorithms for Control of Flight Simulator Motion Systems, AIAA Modeling and Simulation Technologies Conference, Portland, OR, August 9 – 11, 1999, pp. 463 – 473.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37253

AIAA-2000-4292

IN-FLIGHT ESTIMATION OF PILOTS' ACCELERATION SENSITIVITY THRESHOLDS

V V Rodchenko[1]
Central Aerohydraudynamic Institute (TsAGI), Russia
S Yu Boris[2]
Flight Research Institute (FRI), Russia
A D White[3]
Defence Evaluation and Research Agency (DERA), UK

Abstract

A Tu-154M in-flight simulator was used to estimate pilots' sensitivity thresholds for normal, roll and pitch accelerations and to investigate the effect of normal acceleration on angular acceleration sensitivity thresholds. These in-flight experiments were partly reproduced in an on-ground moving-base simulator (FS-102).

Pilot sensitivity thresholds for specific forces and angular accelerations in flight and on the ground were compared. The experiments showed that, due to certain flight factors not normally reproduced on the ground, pilot sensitivity thresholds for normal accelerations in flight can considerably exceed those experienced in ground-based simulators. The experiments also showed that, in ground-based simulators, lateral specific forces arising from cockpit tilt can cause considerable errors in estimating sensitivity thresholds for roll motion.

In-flight experimental data on pilot perception of low-frequency normal accelerations (for frequencies as low as 0.25rad/*s*) and the influence of these accelerations on pilot sensitivity to angular motion were generated. Such low frequency data cannot be obtained using ground-based simulators.

Introduction

Knowledge of acceleration sensitivity thresholds is important for the solution of various motion cueing problems. Examples include identification of the requirements for motion system dynamics, selection of motion-drive algorithms and estimation of the influence of acceleration on piloting[1,2]. Despite the fact that considerable attention has been given to studying acceleration sensitivity thresholds, the issues are not yet fully understood.

Most acceleration perception experiments have been conducted using on-ground moving-base simulators. However, the full flight environment cannot be reproduced adequately on the ground. Moreover, some aspects of acceleration perception can only be studied in flight. This paper presents results obtained using a Tu-154M in-flight simulator.

While studying sensitivity thresholds on the ground such factors as vibrations and sounds are not usually reproduced. These factors may affect kinesthetic and tactile receptors, effectively masking the underlying accelerations and reducing a pilot's sensitivity to them. The influence of these factors on a pilot's acceleration sensitivity can be determined by comparing flight data with ground data. This was one of the goals of the present work.

In the literature, for example references 3 and 5, data on sensitivity thresholds for frequencies greater than 0.7-1rad/s were obtained. Due to platform displacement limitations, data for lower frequencies

[1] Head of Section
[2] Deputy Director of Research Center
[3] Principal Scientist

© British Crown copyright 2000. Published with the permission of DERA on behalf of the Controller of HMSO.
Copyright © 2000 by TsAGI. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

cannot be obtained using ground-based simulators. Thus the second goal of the present work was to determine pilots' sensitivity thresholds for normal accelerations within the low frequency band.

The thresholds of sensitivity to specific forces and angular accelerations are usually studied for separate degrees of freedom. However, in real flight, roll and pitch accelerations act, as a rule, simultaneously with normal specific forces. It was shown in reference 3 that simultaneous normal specific forces can reduce a subject's sensitivity to angular motion. In that work the effect of relatively high-frequency accelerations (ω =1rad/s) on pitch and roll thresholds was considered. Thus, the third goal of the in-flight experiments was to study the effect of low-frequency normal accelerations (ω =0.25rad/sec) on pilots' roll and pitch thresholds.

While investigating sensitivity to pitch and roll accelerations on the ground, longitudinal and lateral specific forces, which are caused by cockpit tilting and are difficult to compensate fully, may lead to errors in determining the thresholds. Thus, the in-flight experiments included tests to evaluate the errors due to false lateral and longitudinal specific forces which could affect the determination of roll and pitch thresholds on the ground.

1. Experimental Facility

Tu-154M In-Flight Simulator (IFS). The IFS (Fig.1) is used to investigate transport aircraft problems. It was developed in 1987 from the Tu-154M 3-engine medium range airliner.



Fig.1- In-Flight Simulator Tupolev-154M

The IFS can reproduce the dynamics and control system of any type of transport aircraft and its dynamic and control system parameters can be varied in flight.

The in-flight simulator has a conventional cockpit with the evaluation pilot in the left seat and the safety pilot in the right seat. The safety pilot has conventional mechanical controls and instruments. He is warned of any failures or system limits and can manually take control of the aircraft by pressing any one of several buttons located on his controls or by moving his control wheel above a certain threshold. The experimental system is disengaged automatically in the event of a system failure or if either the angle of attack or g-load exceeds its limit.

The IFS is equipped with both digital and analog fly-by-wire control systems but in this experiment only the digital control system was used.

The flight data recording system can be programmed to record various flight, control system, engine and other system parameters. In addition, the IFS is equipped with a telemetry up and down link that allows both monitoring and analysis of flight data in real-time and the transmission of ground information (such as control commands, real-time generated HDD/HUD symbology, etc.).

A crew of the Tu-154M IFS normally consists of seven people: evaluation and safety pilots, navigator, project engineer, simulation system engineer and two additional engineers or observers if needed for the purposes of the experiment.

In the course of this particular experiment the following parameters were recorded:

- angular rates for roll p, pitch q and yaw r,
- specific forces along longitudinal n_x, vertical n_z and lateral n_y axes,
- bank ϕ, pitch θ and yaw ψ angles,
- sidestick roll force F_{ox} (not connected to the control system),

  flight indicated speed V_i and altitude H.

The sampling rate was 10Hz.

Angular rate transducers, capable of measuring maximum velocities of ±6deg/sec, were used to measure angular rates in pitch, roll and yaw. Measurement error of the transducers is Δ=1.5%. To measure normal accelerations a transducer with minimum/maximum linear accelerations $-1/+3g$ was used; to measure lateral and longitudinal specific forces, transducers with minimum/maximum accelerations -1/+1 g were used. Measurement error of the transducers is Δ=0.8%.

Linear acceleration transducers are located at the center of gravity. The distance between the transducers and the cockpit is 21 meters. (The maximum error in determining the sensitivity thresholds for normal accelerations at the pilot station are estimated not to exceed 15% even at ω=1rad/s.)

On-Ground Moving-Base Simulator FS-102.
In order to evaluate the effect of flight factors on pilots' acceleration sensitivity, a part of the in-flight

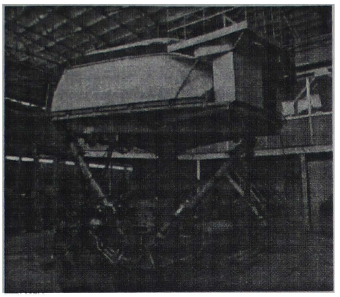


Fig.2- 6DOF Simulator FS-102

experiments was repeated using TsAGI's on-ground simulator (Fig.2). This is a six-degrees-of-freedom synergistic system. The actuators have hydrostatic bearings and a stroke of 1.8m. Maximum values of displacement, velocity and acceleration for each degree of freedom of this motion system are shown in Table 1.

Table 1

	Travel, *m* , *deg*	Velocity, *m/sec, deg/sec*	Acceleration, m/sec^2, deg/sec^2
Surge	±1.75	1.5	7
Sway	±1.23	1.1	8
Heave	±1.475	1.3	7
Roll	±35.1	30	230
Pitch	±37.8	30	230
Yaw	±60	50	260

To measure and record accelerations for all degrees of freedom, six acceleration transducers placed on the simulator platform were used.

The acceleration noise of FS-102 is typical of simulators of this type. For example, Fig.3 illustrates the acceleration noise in heave[3,4]. Also shown is the boundary representing a high level of motion fidelity, in accordance with Ref.2. It can be seen from Fig.3 that the acceleration noise is predominantly below the level of perception.

The position of the subjects' head relative to the rotational axes was 1.6m, 0m and 0.45m along vertical, lateral and longitudinal axes respectively.

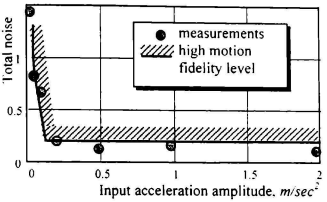


Fig.3 - Motion system acceleration noise in heave (f = 0.5 Hz).

2. Experimental Procedure

Experiments were conducted using sinusoidal accelerations (as used in references 3 and 5).

Three evaluation tasks were performed.

Task 1: Determining the sensitivity thresholds for normal accelerations.

The normal accelerations varied in sinusoidal mode:

$$n_z(t)=A_{nz}(t)\sin\omega t, \qquad (1)$$

Sensitivity thresholds were obtained for normal acceleration frequencies ω = 0.25, 0.5 and 1rad/s.

In every test the amplitude A_{nz} increased slowly, $A_{nz}(t)=c_n \cdot t$, until the pilot was able to sense the direction of motion. Amplitude variation rate, c_n , was $0.005sec^{-1}$ for acceleration frequency ω = 1rad/s, and $0.001se_^{-1}$ for ω = 0.25 and 0.5rad/s.

The pilot deflected the manipulator to register the moment a barely perceptible sensation appeared. As the direction of the motion became clear, the pilot deflected the manipulator as if compensating for the accelerations sensed (though no stick command signals were delivered to the control system). The amplitude of the stick deflections corresponded to the intensity of the pilot's sensations. The acceleration threshold was judged to be the acceleration amplitude at the moment when the pilot started to identify the direction of the motion, which is seen from the time histories presented in Fig.4.

Experiments for ω = 1rad/s were also conducted on the FS-102 ground-based simulator.

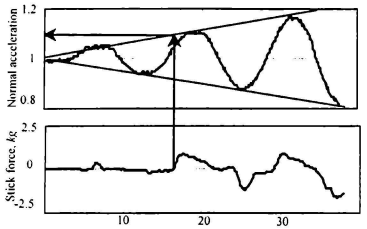


Fig.4 – Determination of the pilot's acceleration thresholds from flight recordings.

Task 2: Determining the roll and pitch sensitivity thresholds.

While determining the pilot's roll sensitivity thresholds the airplane's motion was described as follows:

$$p(t)=A_p(t)\sin\omega t, \qquad (2)$$

The amplitude of sinewaves, A_p , varied linearly as in Task 1; its variation rate, c_p , was 0.2deg/sec in some experiments and 0.135deg/sec in others. The sinusoidal mode frequency was ω=2rad/s.

The value of normal acceleration remained equal to 1g regardless of roll variation.

The procedure to determine the thresholds was the same as in Task 1 (Fig.4).

The sensitivity thresholds for pitch were determined in the same way as for roll, but the pilot was turned in his seat at right angles to the direction of flight, which meant that he perceived roll motion as pitch motion. This procedure was intended to mitigate the effects on pitch perception of any normal accelerations arising from angle of attack variations.

The experiments were reproduced in the ground-based simulator with and without compensation for longitudinal or lateral specific forces caused by cockpit tilting. The specific forces were compensated using linear cockpit motion according to the following equations:

$$\ddot{x}_{cp}=g\,\vartheta,\quad \ddot{y}_{cp}=g\,\phi$$

Task 3: Determining the influence of normal accelerations on angular motion sensitivity thresholds.

The background normal acceleration varied in accordance with (1); the acceleration frequency was ω=0.5rad/s. The background acceleration amplitude was constant: in some tests A_{nz}= 0.25g, in others A_{nz}= 0.25g. (The roll and pitch sensitivity thresholds for zero normal acceleration increment were determined in Task 2.) The angular motion was sinusoidal and varied according to (2); the angular motion frequency was 2rad/s.

The procedure to determine the threshold values was the same as in Task 2.

The flight altitudes were 2000-4000m and flight velocities were 450-500 km/h. Before every test the safety pilot trimmed the aircraft in straight and level flight with constant velocity. Once the aircraft was trimmed the evaluation pilot closed his eyes and the test started within 0.5-1.5minutes. Normal accelerations (equation (1)) and angular rates (equation (2)) were computed by the on-board computer and generated by elevator and aileron deflections. Throttle and pedal positions remained fixed.

The total number of tests was 34, with one sensitivity threshold being determined per test. The duration of each test was between 25 and 120 seconds depending on the input frequency. Between two and four acceleration threshold values were obtained for each configuration considered and these were then averaged.

A single evaluation pilot took part in all in-flight and ground-based experiments. In Task 2, two other subjects also took part in on-ground experiments.

3. Pilot's Normal Acceleration Sensitivity Thresholds

In-flight and ground-based data on pilot's sensitivity to normal accelerations are shown in Fig.5. In this and other figures the average threshold values are shown together with minimum and maximum values obtained in the tests. In-flight data were obtained for acceleration frequencies from 0.25 to 1rad/s, and ground-based data for frequencies from 0.7 to 4rad/s.

The data presented show that, for frequencies above 1-2rad/s, normal acceleration thresholds are almost independent of the acceleration frequency, which corresponds with data presented in other publications[3,4]. As the motion frequency decreases below 1rad/s, however, the pilot's sensitivity decreases

sharply. The threshold doubles as the frequency decreases from 1 to 0.25rad/s.

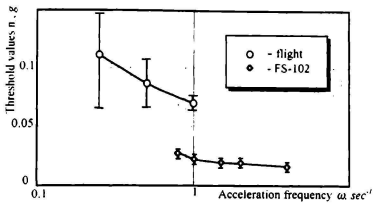


Fig.5 - Normal acceleration threshold values obtained in flight and on the ground.

The comparison of in-flight and ground-based data in Fig.5 for ω=1rad/s shows that the normal acceleration sensitivity thresholds obtained in flight are considerably (2-2.5 times) greater than those obtained on the ground. This difference can be explained by the fact that in the on-ground experiments only normal accelerations were reproduced. A number of flight factors, which can lower pilots' sensitivity to normal accelerations, were not reproduced (as in other ground-based investigations of sensitivity thresholds).

In the literature there are insufficient data on the effect of flight factors on pilots' sensitivity to normal accelerations. We can hypothesise that pilots' sensitivity to normal accelerations is reduced by these factors, which include the following: 1) longitudinal specific forces and pitch accelerations which follow normal accelerations and 2) vibrations and sounds.

The data obtained show that the maximum longitudinal specific forces and pitch rates at the moment the normal acceleration thresholds were registered did not exceed 0.007*g* and 0.4deg/sec respectively. These values do not exceed their respective threshold values, which in accordance with reference 3 are 0.009*g* and 0.8deg/sec (at acceleration frequency ω=1rad/s). Thus, it is unlikely that longitudinal specific forces and pitch rates influenced the pilot's sensitivity to normal accelerations.

It is more likely that the main factors lowering the pilot's normal acceleration sensitivity in flight are vibrations and sounds. It is known that vibration and sound levels can be considerable in flight: according to measurements carried out on similar aircraft, the vibrations within a frequency band from 0 to 10*Hz* are in the range 0.02-0.05*g*.

The vibration and sound environments are similar for different types of aircraft. The acceleration frequency considered, ω=1rad/s, is also characteristic of piloting different aircraft. Thus we can deduce that, for other types of aircraft, pilots' normal acceleration sensitivity thresholds in real flight may exceed those obtained on simulators (in the absence of vibrations and sounds) by a considerable margin.

It should be noted that the criterion to determine the threshold acceleration value used by the pilot in the experiments was somewhat different from the criterion used by subjects in the previous work[3]. According to the pilot, he deflected the manipulator in proportion to his acceleration sensation intensity as if trying to bring it down to zero only when he was convinced he sensed *the direction of the accelerations* correctly. Unlike the pilot, other subjects started to deflect the manipulator in response to *a change in the acceleration direction* rather than to the true acceleration direction itself. For this and/or other reasons (individual sensitivity, inadequate training etc.) the pilot's normal acceleration sensitivity threshold values in the on-ground tests exceeded those of the other subjects (Fig.6).

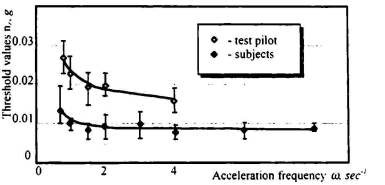


Fig.6 - Differences in pilot and subjects' normal acceleration threshold values.

4. Pilot's Roll and Pitch Sensitivity Thresholds

Fig.7 shows the pilot's roll and pitch thresholds at the angular motion frequency ω = 2rad/s for the three different tests: on the in-flight simulator, on the ground-based simulator with compensation for the cockpit tilting and without it. The data show that if tilt compensation is introduced, roll and pitch thresholds in flight and on the ground are the same. This means that such flight factors as vibrations, sounds, etc. do not affect sensitivity to angular motion.

A question may arise here: why do flight factors affect linear motion perception, yet do not affect angular motion perception? In our opinion, this can be explained by the different roles played by various human sensory systems in the perception of linear and

angular motions[3]. In angular motion perception the vestibular system dominates, whereas in specific forces perception kinesthetic and tactile sensors play a

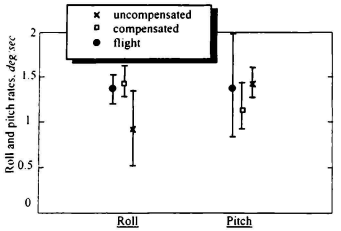


Fig.7 – Roll and pitch threshold values obtained in flight and on ground, $\omega_{p,q}$ =2*rad/s*.

significant role as well (Fig.8). As has been mentioned, vibrations and sounds affect kinesthetic and tactile receptors. Thus, it is natural to suppose that vibrations and sounds dull the specific force perception channel but do not produce any effect on angular motion perception.

sensitivity obtained in simulators without tilt compensation cannot be absolutely correct.

A few questions may arise here. First, do the effects mentioned hold true for other frequency bands? Second, why do the specific forces caused by tilting affect the roll sensitivity thresholds but not pitch sensitivity thresholds? To answer the first question let us consider the roll and pitch threshold values presented in Fig.9, which were obtained for different frequencies with and without tilt compensation. It can be seen that throughout the frequency band considered (from 0.7 to 2rad/s), roll sensitivity thresholds with the tilt compensation exceed the respective threshold values without compensation, while for pitch these threshold values are broadly similar.

To answer the second question let us look at the specific forces arising without tilt compensation at the moment the angular thresholds were registered and compare them with the respective true threshold values. Fig.10 shows data obtained with other subjects. The true lateral and longitudinal specific forces thresholds were described in Reference 3). It can be seen that the lateral specific force values exceed their threshold values by a factor of two. This means that the

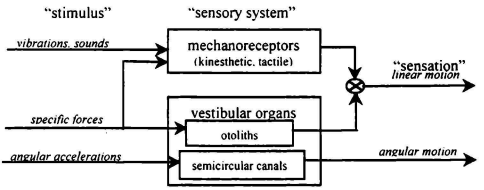


Fig.8 – Differences in generating linear and angular motion sensations

A comparison of in-flight and on-ground "compensated" and "uncompensated" angular motion thresholds (Fig.7) shows two things. First, false longitudinal specific forces due to pitch cockpit tilting do not produce a large effect on pilot's pitch sensitivity thresholds (at least at ω = 2rad/s). Second, false lateral specific forces due to the cockpit tilting in roll can cause considerable errors in determining pilot's roll sensitivity thresholds in ground-based simulators (on-ground thresholds are about 1.5 times smaller than in-flight ones). This means that the data available on pitch sensitivity thresholds are less prone to errors due to false specific forces; the data available on roll sensations caused by roll cockpit tilting (without lateral specific forces compensation) arise due to lateral specific forces rather than angular accelerations. However, what the operators perceive is an angular motion, not a lateral displacement.

Unlike lateral specific forces, longitudinal specific forces at the moment the pitch threshold values were registered are approximately equal to the threshold values of longitudinal specific forces (Fig.10); thus, they do not have a significant effect on the pitch sensitivity thresholds in a simulator without specific force compensation.

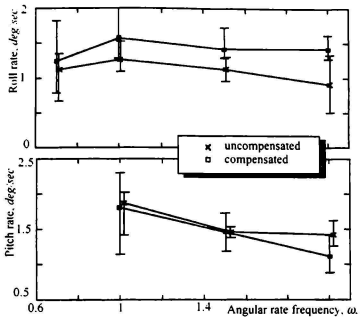


Fig.9 – "Uncompensated" and "compensated" roll and pitch threshold values obtained on ground-based simulator.

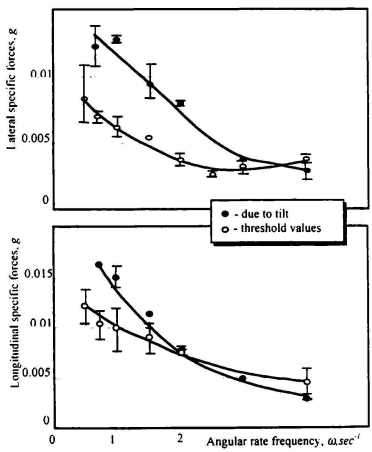


Fig.10- Comparison of false specific forces due to cockpit tilting and the respective threshold values

5. Normal Acceleration Effect on Pilot's Sensitivity to Angular Motion.

Reference 3 showed that the influence of simultaneous linear and angular accelerations dulls the sensitivity to angular accelerations. But in reference 3 only the effect of high-frequency (above 1rad/s) normal accelerations on pilots' sensitivity to roll and pitch motion was considered. Due to the displacement limitations it was impossible to study this effect for lower frequencies on the simulator. For this study the task was to estimate in flight the effect of low-frequency normal accelerations on roll and pitch sensitivity.

Fig.11 shows the data obtained in flight for roll and pitch; the normal acceleration frequency was 0.5rad/s, the acceleration amplitude varied from 0 to 0.5g and the angular motion frequency was 2rad/s. It is clear that normal accelerations have a considerable effect on the roll and pitch threshold values; as the acceleration amplitude A_{nz} increases from 0 to 0.5g, the angular threshold values increase by about 50 %. The roll and pitch threshold values increase linearly in proportion to the same coefficient k=1.5:

$$p = p_0 \left(1 + k \cdot A_{n_z}\right)$$

where p_0 is the threshold value for a particular degree of freedom without vertical acceleration and A_{nz} is the amplitude of the normal acceleration increment related to n_z=1.

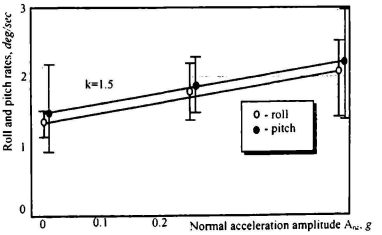


Fig.11 - Effect of normal accelerations on thresholds for roll and pitch.

It should be mentioned that in degree the effect of low-frequency accelerations on angular motion sensitivity differs from the effect of high-frequency (over 1rad/s) accelerations of small amplitudes: coefficient k for relatively small (up to 0.05g) high-

frequency accelerations is greater than for low-frequency accelerations.

CONCLUSIONS

Our in-flight and on-ground tests lead to the following conclusions:

1. Pilots' normal acceleration sensitivity thresholds in normal flight (level flight without atmospheric disturbances) can exceed the threshold values obtained on the ground by a considerable margin. This has been attributed to the presence of vibrations and sounds.
2. As the frequency of motion decreases below 1rad/s, pilots' normal acceleration perception thresholds increase (approximately doubling as ω decreases from 1 to 0.25rad/s).
3. The lateral specific forces due to cockpit tilting have a considerable effect on roll sensitivity thresholds on the ground; data on roll sensitivity obtained in simulators without tilt compensation cannot be absolutely correct.

   The longitudinal specific forces due to cockpit tilting do not produce a significant effect on pitch sensitivity thresholds.

   The roll and pitch sensitivity thresholds obtained on the ground with specific forces compensation introduced appear to be about the same as those obtained in flight. Thus, such factors as vibrations and sounds characteristic of a normal flight do not have a significant effect on pilots' sensitivity to angular motion.
4. Low-frequency (about 0.5rad/s) normal accelerations have a considerable effect on the roll and pitch threshold values: as the accelerations increase from 0 to 0.5g, the angular motion perception thresholds increase by about 50%.

References

1. S.R.Bussolari, R.B.Sullivan, L.R.Young. "Vestibular Models for Design and Evaluation of Flight Simulator Motion", Advanced Flight Simulation – Visual and Motion System Int. Conf. Proceedings, Royal Aeronautical Society, London, 29th-30th April, 1st May, 1986.
2. A.D.White, V.V.Rodchenko. "Motion Fidelity Criteria Based on Human Perception and Performance", AIAA-99-4330, AIAA MST conference, Portland OR, August 1999.
3. L.E.Zaichik, V.V.Rodchenko, I.V.Rufov, Yu.P.Yashin, A.D.White. "Acceleration Perception", AIAA-99-4334, AIAA MST Conference, Portland OR, August 1999.
4. Working Group: "Dynamic Characteristics for Flight Simulator Motion Systems", AGARD Advisory Report No.144, AGARD, NATO, Neuilly sur Seine, France, September 1979.
5. R.J.A.W. Hosman and J.C. van der Vaart. "Thresholds of motion perception measured in a flight simulator." Proceeding of the twelfth NASA-university annual conference on manual control, 1976, USA. p.956-983.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37254

AIAA-2000-4293

MEASURING THE REALISM OF MOTION IN FLIGHT SIMULATORS

Han F.A.M. van der Steen Ph.D.

SIEMENS - The Netherlands
Traffic, Transport and Special Projects division
P.O. Box 16068
NL-2500 BB THE HAGUE
The Netherlands

Abstract

A prominent issue in flight simulation technology is the extent to which inertial (mechanical) motions are necessary. A condition that refers to the need for inertial motions is the perceptual condition that the simulator trainee may not perceive self-movements as unrealistic.

This paper describes results of experiments in a flight simulator which determined inertial motion amplitudes which are needed to provide realistic motion. Two motion directions were tested: roll and yaw.

The experiments demonstrate that the perception of an earth-stationary visual scene corresponds to the perception of realistic self-motion. It is outlined that rather large ranges exist in which visual and inertial self-motion stimuli may physically mismatch, while the simulated self-motion is perceived as realistic.

Introduction

Flight simulators are used to train pilots to handle an aircraft. What kind of simulator is needed for a pilot training depends on the training objectives and the individual training needs (Teunissen 2000). When is known what information processing and actions the trainee can perform before the training and should be able to perform after the training, an appropriate simulator can be selected, such as a Flight Navigation and Procedure Trainer (FNPT), a Flight Training Device (FTD) or a Full Flight Simulator (FFS). When the training objective includes the ability to handle situations in which the pilot's perception of the environment and the perception of self-motion (equal to the aircraft motion) in that environment is required, motion stimuli are needed which induce these perceptions. Motion stimuli can be divided into inertial stimuli and environmental stimuli (Van der Steen 1998). Inertial motion stimuli induce specific forces on the body and the angular accelerations of the body. Environmental motion stimuli are changes of orientations and positions of the environment with respect to the body. The visual perceptual system is the most important sense for the perception of these environmental motion stimuli.

A high-end flight simulator incorporates a synthetic visual system that displays a detailed scene over a large solid angle to enable flight training for visual conditions. This visual system alone can suffice to simulate realistic self-motion in which inertial (mechanical) motion is insignificant or zero, such as self-motion below the perception threshold of inertial motion, or translatory self-motion at a constant velocity. When a larger acceleration needs to be simulated, inertial motions might be needed to render simulated self-motion that is perceived as realistic. Many flight simulators therefore include a motion-base that provides these inertial motions. The primary goal of such a motion-base is to provide sufficient motion workspace, in order to represent the specific forces and angular accelerations of the aircraft (Advani 1998). The range of inertial motions that a motion-base can provide, is limited when compared to aircraft movements. An important issue in motion simulation technology is the extent to which the inertial base motions are actually needed and what criteria for necessity apply. How limited may the inertial motions be while still rendering adequate training, that is, enabling maximum transfer of training for training objectives which are related to

Copyright © 2000 by the authors. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

perceived self-motion? A bottom criterion to measure the need for inertial motion is that the pilot at least may not perceive self-movements as unrealistic. That is, the pilot may not perceive the movement as out of the ordinary. In the present case, the lack of sufficient inertial motion would result in a perceived mismatch of inertial stimulation of the body and visual stimulation of the synthetic environment in which the body is perceived to move.

This paper describes experiments which determined combinations of inertial and visual motions which provide realistic self-motion in a flight simulator. The experiments were part of a research program on motion perception, conducted by three Dutch institutes: TNO Human Factors, Amsterdam Free University, and Delft University of Technology. The relation with earlier motion perception research at TNO Human Factors (e.g. Wertheim 1994) was investigated by relating the perception of an earth-stationary visual scene during self-motion with realistic perceived self-motion through that scene. It is expected that an inertial - visual stimulus area exists in which these stimuli may physically differ from natural congruency while still providing realistic self-motion. This area will be referred to as the "coherence zone".

Experiments

Apparatus

Experiments were conducted using the flight simulator of the Netherlands National Aerospace Laboratory (NLR) in Amsterdam. This simulator included an F-16 cockpit (General Dynamics, USA), a 160 degrees wide visual dome display (Evans & Sutherland, USA) and a six-degrees of freedom mechanical motion system (Hydraudyne, The Netherlands). For details see Van der Steen (1998a, 1998b) or Van der Steen & Brockhoff (2000).

The subject's trunk was strapped in the seat, but the head and limbs were free to move. The subject wore a head-set for communication. In addition, the sound of the simulator motion base was masked by providing 90 dB white noise through the head-set.

Method

The subjects (n = 6) initiated simultaneous base motion and visual scene motion by pulling the side-stick. A test trial consisted of inertial and visual motion with a total duration of three seconds. Both the inertial and visual motion had a motion profile as illustrated in Figure 1. These motions were short acceleration-deceleration profiles that regularly occur both during natural body movements and during body motion in aircraft.

The subject had a dual response task. After each trial, the subject indicated if the visual scene had remained earth-stationary and the subject indicated if the motion was perceived as realistic. The subject pressed a side stick button if the visual scene was perceived as not earth-stationary in a trial. The subject verbally reported to the experimenter whether the motion was realistic or not.

The time histories of the stick deflection, the visual and inertial motions, and the button presses were registered. The response of "no button press" was determined by the absence of a button press between the end of stimuli presentation and the next stick deflection.

Experimental sessions were conducted that determined the lower and upper inertial threshold amplitudes at each visual scene acceleration amplitude. The lower and upper inertial thresholds were defined as the inertial motion amplitudes for which the inertial stimulations were too small or too large, respectively, to provide the perception of an earth-stationary visual scene.

The visual stimulus had a constant acceleration amplitude at either 0, 2, 4, 8, or 12 deg/s^2. A staircase procedure varied the inertial stimulus acceleration amplitude over trials and determined the lower and upper inertial thresholds of the range in which the visual scene was perceived as earth-stationary. The amplitude of the first inertial stimulus in the staircase was randomly applied between 0.8 and 1.2 times the visual scene

acceleration amplitude to avoid identical staircase histories from different sessions. The initial staircase step size was chosen at 20 per cent of the visual scene acceleration amplitude.

The staircase procedure was continued until the step size became less than one-eighth of the initial step size. The session was also stopped when the number of trials exceeded 30 to avoid fatigue in the subject. A short break of about half a minute was taken between two sessions. Each threshold was measured three times for every subject. An additional staircase was conducted, however, when the total amount of trials from one condition was below 20. The lower and upper inertial thresholds were determined from all response data that were obtained at each visual scene acceleration amplitude tested.

Two types of motion were tested: roll motion about the longitudinal body axis and yaw motion about the vertical body axis. Thresholds for both motion directions were determined in separate sessions.

Results

The solid circles in Figure 2 depict the lower and upper inertial thresholds that were determined in the experiments. The solid lines depict the regression lines of the lower and the upper inertial thresholds. The dashed lines indicate the physically congruent visual and inertial stimulation. The open circles are hypothetical thresholds, assuming symmetry around the zero visual amplitude axis.

The subjects verbal responses allowed for a categorization into three classes: "the motion was not realistic", "something was out of the ordinary", and "the motion was realistic". Comparison of the verbal reports with the button press data showed that when the motion was perceived as not realistic, the scene was perceived as not earth-stationary. The inertial motions at which the subjects reported that "something was out of the ordinary", had amplitudes around the inertial thresholds. The combinations of the remainder of visual and inertial motions between the lower and the upper inertial thresholds were perceived as realistic and the visual scene as earth-stationary.

Discussion

A relatively large acceleration amplitude range exists in which inertial body motion and visual scene motion physically differ from congruency while the self-motions are perceived as realistic. The inertial acceleration threshold amplitudes that bound this range, the lower and the upper inertial thresholds, can be approximated by linear functions of the visual scene acceleration amplitude. The condition of an earth-stationary perceived visual scene corresponds to realistic simulated self-motion.

The coherence zone is likely to depend on several task and stimulus related parameters. The subject's attention, for example, was focused solely on detecting movement of the visual scene. It can therefore be inferred that the differences between the lower and upper inertial threshold amplitudes are the smallest that can be found for the motions tested and the set-up used. When the subject performs a control task or a distracting task, for example, the magnitude of the area between the lower and upper inertial thresholds amplitude curves might be larger. The coherence zone is also likely to depend on the visual scene layout. The threshold curves may be closer when the scene contains fewer detail and natural objects. Such visual scenes have a lesser effect on perceived self-motion than scenes containing more natural features (Howard and Childerson 1994). When adding depth and more resolution and natural objects to the visual scene tested here, on the other hand, the influence of the visual scene on perceived self-motion can be larger, and the threshold curves may be more diverted. As a result, the magnitude of the area between the lower and upper inertial threshold amplitude curves may well be a measure for the "optokinetic power" or "compellingness" of an artificial visual scene and virtual environments in general. The larger the area between the threshold curves, the more optokinetic or compelling the visual scene is. Consequently, less inertial motion

may be need to achieve realistic simulated self-motion.

Due to the limited excursion of motion bases, the knowledge of visual-inertial coherence zones is essential in order to generate combinations of visual and inertial stimulations for simulated self-motion through a visual scene that is perceived as realistic. The lower inertial thresholds determined can be applied to define the least inertial stimulations necessary for realistic simulated self-motion while the upper inertial thresholds, especially around the 0 deg/s^2 condition, can be applied for a desired return of the simulator base to the mid position. In general, the coherence zone found here, can be applied for the design of a perception model that predicts realistic simulated self-motion. Such a model is an extension of the classical motion base algorithms that are typically based on vestibular dynamics (Advani 1998).

Concerning the issue of the need for inertial motion: It can be inferred that inertial motion should not exceed the boundaries of the coherence zone (see Figure 2) when realistic simulated self-motion needs to be achieved. It should be noted, however, that substantial inter-individual differences may exist. Furthermore, the criterion for perceived realism to measure the need for inertial motion seems like a bottom criterion for inertial stimulation. Other criteria might concern, for example, the skill-based level instead of the cognitive level. Experiments showed (Hosman 1996) that inertial motion feedback improves performance and changes skill-based control behavior in tracking tasks. It would be interesting to apply the knowledge from perceptual coherence zones to, perhaps, coherence zones from skills tasks, such as tracking tasks, which could be defined as areas in which the performance exceeds a certain level.

Acknowledgements
The author wishes to thank the Netherlands National Aerospace Laboratory (NLR) for providing access to the flight simulator and for their assistance during the experiments. Furthermore, the author wishes to thank the Royal Netherlands Airforce (RNLAF) for sponsoring this research.

References

Advani SK (1998). The kinematic design of flight simulator motion-bases. Ph.D. Thesis. Delft University of Technology. ISBN 90-407-1671-4.

Hosman RJAW (1996). Pilot's perception and control of aircraft motions. Ph.D. Thesis. Delft University of Technology. ISBN 9040713847.

Howard IP & Childerson L (1994) The contributions of motion, the visual frame, and visual polarity to sensations of body tilt. Perception vol. 23; pp. 753-762.

Steen FAM van der (1996). Simulating self-motion. Brain Research Bulletin vol. 40; pp. 473-475.

Steen FAM van der (1998a). Self-motion perception. Ph.D. Thesis. Delft University of Technology. ISBN 9090117571.

Steen FAM van der (1998b). An earth-stationary perceived visual scene during roll and yaw motions in a flight simulator. Journal of Vestibular Research vol. 8(6); pp. 411-425.

Steen FAM van der & Brockhoff PTM (2000). Induction and impairment of saturated yaw and surge vection. Perception & Psychophysics vol. 62(1); pp. 89-99.

Teunissen RPAM (1999). The future of simulation and training. In: "Can flight Simulation do everything?". Proceedings of the Royal Aeronautical Society Flight Simulation Group Conference, London, 19-20 May 1999.

Wertheim AH (1994). Motion perception during self-motion: The direct versus inferential theory revisited. Bevahioural Brain Sciences vol. 17; pp.293-355.

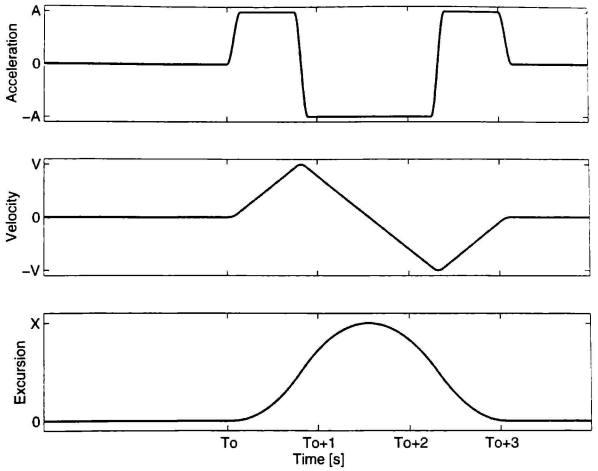


Figure 1: The visual and inertial stimulus motion profile of one trial, showing the acceleration (top), velocity (middle) and excursion (bottom). Following stick deflection by the subject at t = To, an acceleration amplitude A was applied for 0.75 s, followed by a deceleration of −A for 1.50 s and an acceleration A for 0.75 s. A quarter of a period of a $\cos^2(\omega t)$ function with $\omega = 10.5$ rad/s was applied to smooth the acceleration steps in order to prevent the presentation of high motion frequencies to the simulator base. Reprinted from Van der Steen (1998b), Copyright 1998 with permission from IOS Press, Amsterdam.

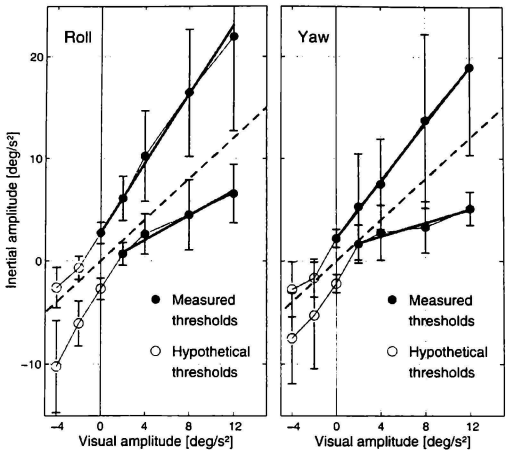


Figure 2: The mean group lower and upper inertial thresholds and standard deviations from the roll and yaw experiment. The solid circles indicate the measured thresholds. The open circles represent hypothetical thresholds. The dashed line indicates physically congruent stimulation of equal visual and inertial acceleration amplitudes. The thick solid lines are the weighted least squares linear regression lines of the thresholds. Reprinted from Van der Steen (1998b), Copyright 1998 with permission from IOS Press, Amsterdam.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37255

AIAA-2000-4295

A THEORETICAL APPROACH TO ESTIMATION OF ACCELERATION EFFECTS ON PILOTING

L E Zaichik[1], V V Rodchenko[2], Yu P Yashin[3], I V Rufov[4]
Central Aerohydrodynamic Institute (TsAGI), Russia
A D White[5]
Defence Evaluation and Research Agency (DERA), UK

Abstract

The paper presents an analysis, based on experimental data, of the effects of acceleration on pilot ratings and performance for various piloting tasks and aircraft characteristics. These acceleration effects are discussed and a theoretical approach to the estimation of these effects for different aircraft and control axes is proposed and substantiated. The effectiveness of the approach is illustrated by an analysis of the effects of acceleration on roll control.

Introduction

It is well known that accelerations can have a considerable influence on piloting. Therefore knowledge of these effects is necessary to solve many flight simulation problems, for example to estimate the need for motion platform cueing for a particular task or to correct experimental results obtained on fixed-base simulators. Such knowledge is also required to improve existing manual control theory and handling qualities criteria.

Previous literature available on the subject compare data for fixed-base simulations with moving-base or in-flight simulations[1-5]. The data demonstrate that motion can have a considerable effect on the control process: the effect can be beneficial, negative or, in some cases, zero, depending on particular aircraft characteristics and types of flight task. Due to the complex nature of these acceleration effects, it is difficult to predict their effects and experiments are usually required in each case to estimate handling qualities. Thus an approach is required to integrate various experimental data and create a theory which can be applied generally.

Recently, such a theoretical approach was developed to provide a method to estimate the effects of acceleration in different control axes for different piloting tasks and aircraft characteristics. The theory was outlined in Ref.6 and is described in greater detail and developed further in the present paper.

The fundamental principles of the approach are based on the notion of the pilot as a link in a control loop and on knowledge of human acceleration perception. Thus, the principles hold true for different control axes. However, the mathematics and empirical functions used to apply the principles quantitatively may differ for different control axes due to differences in longitudinal and lateral vehicle dynamics, piloting tasks, the pilot's position relative to the rotational axes and acceleration perception for the different degrees of freedom. This paper applies the principles to estimate the effects of accelerations on roll control.

1. General Description of the Approach.

According to the proposed approach, the role of accelerations in piloting is twofold (Fig.1). On the one hand, they provide motion cues which are beneficial for piloting (information factor). On the other hand, accelerations produce effects which can

[1] Leading Research Engineer
[2] Head of Section
[3] Leading Research Engineer
[4] Engineer
[5] Principal Scientist

© British Crown copyright 2000. Published with the permission of DERA on behalf of the Controller of HMSO.
Copyright © 2000 by TsAGI. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

be unpleasant and felt by the pilot to have a negative influence (physiological factor). The net effect of accelerations (negative, positive or neutral) in any particular case depends on the dominant factor for that case.

Information (beneficial) factor. As a result of acceleration effects on the pilot, one or more motion feedbacks are added to the visual feedback(s) in the pilot–aircraft system (Fig.2). Thus the effect of the

Physiological (negative) factor. Quite large g-loads may not be perceived to be negative in cases where these g-loads arise as a result of a pilot's control activity. However, even relatively insignificant accelerations may be felt to be unpleasant if they do not result from, or are not correlated with, the pilot's activity.

For some types of vehicle (e.g. those with a short roll mode time constant) the aircraft can respond to the

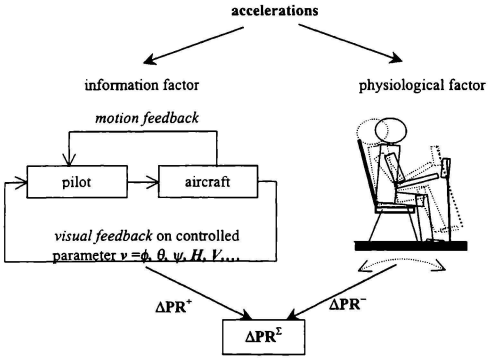


Fig.1 - Factors influencing the effects of acceleration on piloting.

information factor on piloting can be evaluated in the same way as the effect of feedback on control system performance.

To estimate the effect of feedback on the performance of any control system we need to analyze a dynamic model of the system (see Fig.2, for example). Since aircraft dynamics is the main component in the control loop, whatever the piloting task may be, the motion effect inevitably depends on aircraft dynamic characteristics.

The structure of the pilot-aircraft system depends on *v(t)*, which is the visually controlled parameter in a particular case. The controlled parameter is selected according to the piloting task; for example, it may be bank angle, roll tracking error, glide path parameters, vertical velocity, etc. This means that the structure of the pilot-vehicle system, and consequently the influence of accelerations in piloting for different piloting tasks, can differ even if the aircraft characteristics are the same.

high-frequency component of a pilot's control input, an effect which is exaggerated if the pilot's height relative to the rotational axis is large. As a result, accelerations with frequencies beyond the normal piloting frequency band may arise. These cause involuntary body movements, since high-frequency disturbances make proper positioning of the body very difficult to maintain. The effect of high-frequency specific forces is not only unpleasant but can interfere with visual perception and control movements. All these effects contribute to negative pilot opinion of aircraft handling qualities.

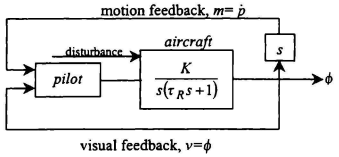
Combined effect of the two factors. The technique proposed in this paper evaluates the effect of acceleration on piloting according to differences in pilot ratings for fixed-base simulation PR_f, on the one hand, and moving-base simulation PR_m or real flight, on the other. As the acceleration effect on piloting depends on the information and physiological factors

acting simultaneously, the difference in pilot ratings is as follows:

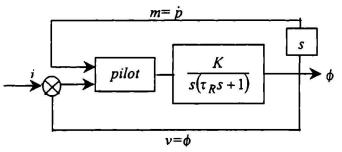
$$PR_m - PR_f = \Delta PR^- - \Delta PR^+ \qquad (1)$$

where ΔPR^- and ΔPR^+ are the differences in pilot ratings due to physiological and information effects respectively.

Roll control (disturbance task)



Roll control (tracking task)



Pitch control for conventional aircraft

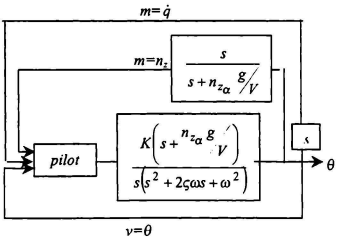


Fig.2 - Pilot-aircraft system models for different piloting tasks.

2. Estimation of the Beneficial Acceleration Effects.

Acceleration effects in different piloting tasks. The motion feedback loop (pilot - aircraft - accelerations) is the same for any piloting task, since the accelerations resulting from pilot control activity are determined by aircraft characteristics only. Unlike the motion feedback loop, the structure of the visual feedback loop (pilot - aircraft - observed parameter) also depends on the piloting task. Thus, the visual-to-motion cues transfer functions $Y_{m/v}$ can differ for different piloting tasks (Fig.2). The paper will show how the form of these transfer functions determines the beneficial acceleration effect for a particular piloting task.

The influence of the information factor in a particular task is determined by the amplitude-phase relationship between motion and visual cues. Analysis of experimental data shows that, where accelerations are the second derivatives of the visually controlled parameters (for example, for bank angle and altitude control tasks, for pitch control in VTOL hovering, etc.), motion cues affect pilot opinion beneficially. If accelerations differ markedly from the second derivative of the controlled parameter, e.g. due to phase lag or lead introduced by motion platform actuator dynamics and driving laws, the beneficial motion effect disappears. This, together with other experimental findings, allows us to formulate a criterion to evaluate the beneficial motion effect for different piloting tasks ($Y_{m/v}$ criterion): the effect can be beneficial only if accelerations approximate the second derivatives of the visual parameter being controlled by a pilot, within the piloting frequency range:

$$m = \ddot{v} \qquad (2)$$

The effectiveness of the criterion can be illustrated if we compare the acceleration effects in disturbance and tracking tasks for roll control[7,8]. Fig.2 shows that the structures of the pilot-aircraft system for these tasks are different. For disturbance tasks, accelerations are the second derivative of the controlled parameter:

$$\dot{p} = \ddot{v} = \ddot{\phi}\,.$$

It follows from the criterion that accelerations can produce a beneficial effect on piloting for a disturbance task.

According to McRuer's theory[9]. the open loop describing function of a pilot-aircraft system for a

tracking task near the crossover frequency, ω_c, is as follows:

$$Y_{pilot} Y_c = \omega_c / s \; e^{-s\tau}$$

Thus, for a tracking task at about the crossover frequency we have the following expression for roll accelerations:

$$\dot{p} = \omega_c \cdot e^{-s\tau} \cdot \dot{v}$$

where v is $v(t)=i(t)-\phi(t)$ for a tracking task.

It follows from this that accelerations are not the second derivative of the controlled parameter v for a tracking task. Thus, it follows from the $Y_{\dot{m}/v}$ criterion that accelerations do not produce any beneficial effect on piloting.

This theoretical conclusion is in agreement with various experimental data[8,9], which increases confidence in the validity of the criterion for evaluating the beneficial effect of accelerations in different piloting tasks.

Acceleration effects for different aircraft characteristics. The beneficial effect of accelerations, when they satisfy Eq.(2), is due to the fact that motion perception is direct and is practically independent of pilot workload. As a result, the pilot's time delay diminishes and the pilot-aircraft system stability margin increases (Fig.3). The pilot can therefore increase his gain and improve piloting accuracy and pilot ratings.

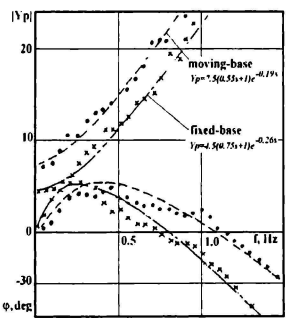


Fig.3 - Pilot describing functions in fixed-base and moving-base simulations (disturbance task).

The effect of acceleration is similar to the effect of time delay in an aircraft control system. Therefore, to estimate acceleration effects we can take advantage of methods used to estimate the influence of aircraft control system delay on handling qualities. A number of criteria are available, one of which is the bandwidth criterion which is both simple and applicable to different control axes and aircraft types[10]. The bandwidth parameter has therefore been adopted to estimate the beneficial effects of accelerations.

The method to estimate beneficial acceleration effects on roll control. In this paper the approach is applied to roll control.

Angular accelerations arising during roll control satisfy Eq.(2). Thus motion cues in this case play a beneficial role.

To evaluate quantitatively the beneficial acceleration effects, ΔPR^+, we use the bandwidth parameter ω_{BW}. Fig.4 demonstrates that roll-axis experimental data can be approximated adequately by the following expression for ΔPR^+:

$$\Delta PR^+ = \begin{cases} 2, \omega_{BW} < 1.5\,\text{sec}^{-1} \\ 2.55 - 3.14 \cdot \lg \omega_{BW}, \; 1.5 \le \omega_{BW} < 6 \\ 0, \omega_{BW} \ge 6\,\text{sec}^{-1} \end{cases} \quad (3)$$

where ω_{BW} is defined by the following condition:

$$\varphi\left[Y_{p/\delta as}\left(j\omega_{BW}\right)\right] = -45^o .$$

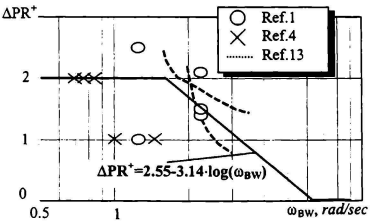


Fig.4 - Beneficial acceleration effect vs bandwidth, ω_{BW}.

Roll control is an indispensable element for all tasks requiring aircraft lateral motion control. Thus roll accelerations play a beneficial role in practically all tasks, provided Eq.(2) holds. The beneficial effect of motion for low-bandwidth aircraft (τ_R>0.45sec) is confirmed in Fig.5, which compares data from fixed-base experiments (top) with flight (middle); the data

are averaged over 8 different tasks of lateral motion control[1].

3. Estimation of the Negative Acceleration Effects.

The reasons for negative acceleration effects were considered in section 1. In this section the method is applied to estimate acceleration effects in the roll axis.

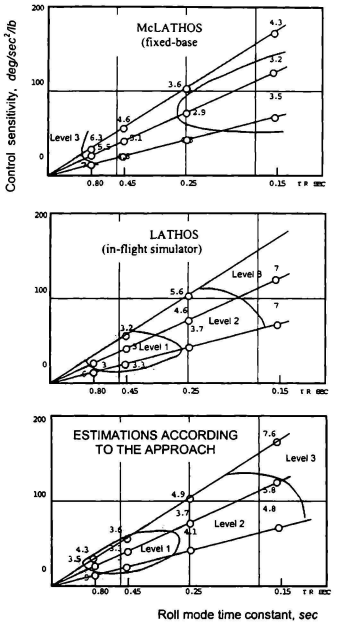


Fig.5 - The comparison of fixed-base and flight test data with the estimations according to the approach proposed.

The main principles of the method. It is seen from Fig.5 that roll handling qualities in flight degrade as the roll mode time constant decreases[1] (τ_R <0.4 *sec*). This is due to the fact that at these roll mode time constant values the aircraft responds to the high-frequency component of the pilot's control activity, and, as a result, high-frequency angular accelerations arise. The effect is illustrated in Fig.6, which presents angular acceleration spectra for τ_R=1 and 0.1*sec*. These high-frequency angular accelerations cause lateral specific forces:

$$n_y = \frac{h}{g} \cdot \dot{p} ,$$

where h is the height of the pilot's position relative to the rotational axis. These lateral specific forces are the cause of the pilot's negative opinion of the aircraft handling qualities.

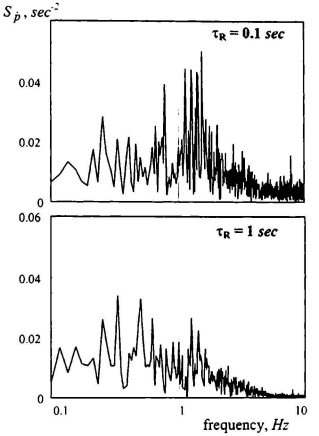


Fig.6 - Spectra of roll accelerations in tracking task for different τ_R .

Fig.7 presents experimental data on human perception of lateral specific forces in the presence of background roll motion. The figure shows that lateral acceleration sensitivity thresholds increase in proportion to roll rate. This fact enables us to assume that the extent to which pilot ratings are affected by the high-frequency lateral accelerations depends on the ratio (λ) between the values of lateral acceleration and roll rate:

$$\Delta PR^{-} = \Delta PR^{-}(\lambda), \qquad (4)$$

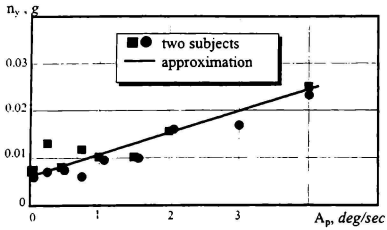


Fig.7 - Lateral acceleration thresholds vs roll rate amplitude.

The lateral accelerations and roll rates can be described by their root-mean-square values σ_{ny} and σ_p. Therefore,

$$\lambda = \frac{\sigma_{n_y}}{\sigma_p} .$$

Fig.8 shows the experimental relationship between pilot rating increments ΔPR^- and parameter λ. The data was obtained using a moving-base simulator to reproduce aircraft motion with various values of roll mode time constant and pilot height. Function (4) can be approximated by the following expression:

$$\Delta PR^-(\lambda) = \begin{cases} 0, \lambda < 0.2 \\ 9 \cdot \log(\lambda) + 6.3, \lambda \geq 0.2 \end{cases} \qquad (5)$$

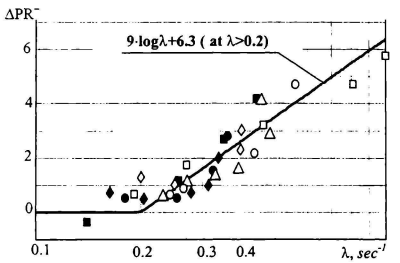


Fig.8 – Negative acceleration effect vs parameter λ.

A method to estimate λ. To estimate parameter λ we need to have a pilot behaviour model.

Pilot behaviour can be assumed to consist of a useful low-frequency component and a high-frequency (noise) component. In general, the spectrum characteristics of the noise component and, consequently, the negative acceleration effects, depend on the aircraft characteristics and piloting task. To estimate negative effects it is natural to consider the worst case. Experiments have shown that negative acceleration effects are especially noticeable when a pilot tries to compensate for small low-frequency accelerations. In this case lateral accelerations and roll rates are determined mainly by the noise component of pilot behaviour.

Investigations have shown that, to a first approximation, the noise power spectrum referred to the noise root-mean-square value is virtually unaffected by manipulator type or external disturbances. It is also unaffected by the values of roll mode time constant and prefilter time constant provided these values are small enough (a necessary condition for the negative effects to occur). The noise component can be represented by white noise passing through a first order filter as follows:

$$Y_{pilot} = \frac{1}{s + \omega_c \cdot \frac{K_p}{K_{p*}}} , \qquad (6)$$

where ω_c is the pilot-aircraft system crossover frequency for $K_p = K_{p*}$ (the optimum roll sensitivity), $K_p = Y_{\left(p/\delta_{as}\right)}(j\omega)\big|_{\omega=0}$ i.e. the steady-state gain in the stick to roll rate transfer function $Y_{\left(p/\delta_{as}\right)}(j\omega)$. This model of the noise component agrees with published data, e.g. Refs.11, 12. Model (6) differs from previous models due to the fact that the relationship between crossover frequency and roll sensitivity has been revealed in the present work.

Model (6) is, in fact, a random law of manipulator deflection. According to well-established expressions from random functions theory, the values of parameters σ_{ny} and σ_p can be determined by the following expressions:

$$\sigma_{n_y}^2 = \frac{h^2}{g^2} \cdot \frac{1}{2\pi} \cdot \int_{-\infty}^{+\infty} \omega^2 \left| Y_{\left(p/\delta_{as}\right)}(j\omega) \cdot Y_{pilot}(j\omega) \right|^2 d\omega$$

$$\sigma_p^2 = \frac{1}{2\pi} \cdot \int_{-\infty}^{+\infty} \left| Y_{\left(p/\delta_{as}\right)}(j\omega) \cdot Y_{pilot}(j\omega) \right|^2 d\omega \qquad (7)$$

If we know the roll transfer function or its magnitude, integrals (7) can be found numerically. For functions $Y_{p\,\delta as}(j\omega)$ up to fifth order the solutions for integrals (7) are given in reference books.

For many problems in handling qualities and flight simulation, the roll transfer function can be defined by first or second order models:

$$\frac{p}{\delta_{as}} = \frac{K_p}{\tau_R s + 1}$$
$$\frac{p}{\delta_{as}} = \frac{K_p}{\tau_R s + 1} \cdot \frac{1}{Ts + 1} \cdot e^{-s\tau} \qquad (8)$$

where T is prefilter time constant and τ is equivalent time delay.

Since the first transfer function in (8) is a particular case of the second transfer function, we use the second transfer function in (8) to define the function λ of aircraft characteristics. Thus the function in question is as follows:

$$\lambda = \frac{h}{g} \cdot \sqrt{\frac{\omega_c \cdot K_p / K_{p^*}}{T + \tau_R + T \cdot \tau_R \cdot \omega_c \cdot K_p / K_{p^*}}} \qquad (9)$$

Comparison of experimental data with estimates derived from Eq.(9) for different values of ω_c and K_{p^*} has shown that the best agreement is achieved if ω_c=2 rad/*sec* and K_{p^*}=0.025 *rad/sec/mm*.

The proposed approach to estimation of beneficial and negative acceleration effects produces results which agree with a variety of flight and ground-based test results for conventional aircraft, helicopters and VTOL aircraft. Good correlation between the estimates and experimental data has been achieved, as shown in Fig.5 for example for different roll mode time constants and control sensitivities.

Conclusions

Accelerations can have a considerable influence on pilot control activity and a pilot's opinion of aircraft handling qualities. The effects can be beneficial or negative. They may be beneficial because they give the pilot additional cues that lead the visual cues. The negative effect manifests itself as the physiological factor due to high-frequency accelerations at small values of roll mode time constant. The net effect is determined in each case by the aircraft dynamics and the piloting task.

Methods to estimate the acceleration effect for different aircraft types and dynamic characteristics have not been sufficiently well developed to date. This paper has described an approach based on analysis of the perception process and the mechanism by which motion cues influence pilots. The approach is applicable to all aircraft control axes.

The paper has also shown how the method can be applied to estimate the effect of roll acceleration on the handling qualities of different aircraft. Estimates based on this method agree with various experimental data.

References

1. J.R.Wood. "Comparison of Fixed-Base and In-Flight Simulation Results for Lateral High Order Systems". AIAA-83-2105, August 15-17, 1983, Gatlinburg, Tennessee.
2. Schroeder, J.A., Chung, W.Y., Tran, Duc T., Laforce, S., Bengford, N.J., "Pilot-Induced Oscillation Prediction With Three Levels of Simulation Motion Displacement", AIAA Paper 98-4333, 1998.
3. Brandon, J.M.., Glaab, L.J., Brown, Ph.W., Philips, M.R., "Ground-to-Flight Handling Qualities Comparisons for a High Performance Airplane", AIAA Paper 95-3457, Baltimore, MD, 1995.
4. A.D.White, J.R.Hall, B.N.Tomlinson. "Initial Validation of an R&D Simulator With Large Amplitude Motion". AGARD-CP-513.
5. V.V.Rodchenko, L.E.Zaichik, Yu.P.Yashin. "Motion Cueing on Research Flight Simulators". CEAS Symposium on Simulation Technology "Making it Real", Delft, The Netherlands, 30,31 Oct. - 1 Nov., 1995.
6. A.D.White, V.V.Rodchenko. "Motion Fidelity Criteria Based on Human Perception and Performance". AIAA-99-4330, AIAA MST Conference, Portland, OR, August 1999.
7. R.J.A.W.Hosman "Visual-Vestibular Interaction in the Perception and Control of Aircraft Motions by the Pilot". AIAA Modeling and Simulation Technologies Conference, New Orleans, August, 1997.
8. Efremov, A.V., Rodchenko, V.V., Boris, S.Yu., et al. "Investigation of Pilot-Induced Oscillation Tendency And Prediction Criteria Development". WL-TR-96-3109, Wright-Patterson AFB, Feb 1994 – March 1995.
9. McRuer D., "Human Dynamics in Man-Machine Systems'. Automatica., vol.16, No.3, 1980, pp.237-253.

10. Chalk C.R. Background Information and User Guide for MIL-F-8785B Military Specification Flying Qualities of Piloted Airplanes. AFFDL-TR-69-72, 1969.
11. A.V.Efremov, V.V.Rodchenko, A.V.Ogloblin, A.N.Predtechenky. "Pilot As a Dynamic System". Mashinostroeniye, 1992, 336p. (in Russian).
12. D.M.McRuer, E.Krendel. "Mathematical Models of Human Pilot". AGARD-AGD-188, 1974, 72p.
13. D.G.Mitchell, R.H.Hoh, J.M.Morgan, "Flight Investigation of Helicopter Low-Speed Response Requirements"' Journal of Guidance, Control, and Dynamics, vol.12, No.5, Sept.-Oct. 1989, pp.623-630.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37256

AIAA-2000-4296

THE EFFECT OF SIMULATOR MOTION ON PILOT TRAINING AND EVALUATION*

Tiauw H.Go†
Massachusetts Institute of Technology, Cambridge, Massachusetts

Judith Bürki-Cohen‡
Volpe Center, U.S. Department of Transportation,
Cambridge, Massachusetts

Nancy N. Soja§
Brookline, Massachusetts

ABSTRACT

This study empirically examined the effect of simulator platform motion on airline pilot recurrent training and evaluation. It is driven by the need for sound scientific data on the relationship between certain key modern device features and their effect on the transfer of pilot performance and behavior to and from the respective airplane. The experiment utilized an FAA qualified Level C simulator with six-degree-of-freedom synergistic motion and a wide angle high quality visual system. Experienced airline pilots were evaluated and trained in the simulator, half of them with and the other half without motion. Then the transfer of skills acquired by both groups during this training was tested in the simulator with the motion system turned on as a stand-in for the airplane (quasi-transfer). Every effort was made to avoid deficiencies in the research design identified in a review of prior studies, by measuring pilot stimulation *and* response, testing both maneuvers and pilots that are *diagnostic* of a need of motion, avoiding pilot and instructor *bias*, and ensuring sufficient statistical *power* to capture operationally relevant effects. The results of the analyses as well as their implications are presented in this paper.

NOTATIONS

FAA	Federal Aviation Administration
PTS	FAA Practical Test Standards
RTO	Rejected Take-Off
V_1	Take-off decision speed; the minimum speed in the take-off, following a failure of the critical engine, at which the pilot can continue the take-off and achieve the required height above the take-off surface within the take-off distance.
V_1 cut	Engine failure at or above V_1 with continued take-off
V_2	Take-off safety speed; a speed that will provide at least the gradient of climb required by the airplane certification rules with the critical engine inoperative.
PF	Pilot Flying
PNF	Pilot Not Flying
I/E	Instructor/Evaluator
n	Sample size
p	Probability of null hypothesis (i.e., no effect of motion)
r	Pearson correlation coefficient
STD	Standard Deviation

INTRODUCTION

This research effort is part of the Federal Aviation Administration's (FAA) initiative towards promoting the availability and affordability of flight simulators for U.S. commuter airline training.[2] This initiative becomes even more important as the FAA is proposing a rule that would mandate the use of simulators for all air carrier training and qualification, limiting the use of the aircraft itself as a training option even for small regional airlines. However, there is a lack of sound scientific data on the relationship between

* Portions of this work have been published as Ref. 1.
† Postdoctoral Associate, Department of Aeronautics and Astronautics, Member AIAA.
‡ Engineering Psychologist.
§ Consultant, experimental psychology.

This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

certain key training device features, such as platform motion cuing, and their effect on the transfer of performance to and from the airplane. This project will develop a scientific basis to assure that FAA requirements promote full transfer of pilot performance between simulator and airplane without unnecessarily driving up cost. The data will also help the FAA to evaluate air carrier proposals for the alternative use of other training equipment in lieu of full flight simulators without compromising safety objectives. The first stage of this multi-year project was a state-of-the-art review of key aspects of flight simulation, involving both FAA and Industry subject matter expert workshops[3,4] and an extensive literature review.[5,6,7] Based on this review, an empirical investigation of flight simulator requirements which seeks to correct deficiencies in the research design of prior studies has been initiated.

The present study empirically examined the effect of platform motion (i.e., FAA qualified Level C six-degree-of-freedom synergistic motion) in the presence of a high-level visual system (i.e., wide-angle collimated cross-cockpit) on pilot training and pilot evaluation. It addressed the questions of whether the motion provided by an FAA qualified Level C simulator affects 1) First Look evaluation of pilot performance and behavior prior to any simulator practice, 2) the course of Training in the simulator, and 3) the Transfer of training acquired during training in the simulator with or without motion to the simulator with motion as a stand-in for the airplane. The analysis also examined whether the grading criteria used by the instructors/evaluators (I/Es) were affected by the presence or absence of motion. The resolution of the experiment was also considered, i.e., the power to find the effect of motion if there is one.

RESEARCH METHOD

The experiment used an FAA qualified Level C flight simulator, which represents a 30 passenger, three crew, turboprop airplane with wing-mounted twin engines and counter-rotating propellers. The six degree-of-freedom synergistic motion system has hydraulically actuated legs capable of a 60 inch stroke. The high quality visual system provides wide angle collimated cross-cockpit viewing with a 150 degrees horizontal and 40 degrees vertical field of view available to each pilot.

The research was conducted with regional airline pilots in recurrent training. Data were collected from 42 crews. Two experiments were combined into one experimental session in order to minimize the disruption to the host airline's training and evaluation program, as well as to reduce pilot adaptation to a simulator configuration. Both experiments investigated the need for platform motion in simulators, focusing on different functions of the simulator. The first experiment, First Look evaluation, examined the use of simulators as *evaluation* tools of pilots' aviating skills. In other words, it assessed the degree to which a pilot's existing skills transferred *from* the airplane *to* the simulator, and whether this was affected by the motion state of the simulator. This assessment needed to occur during the very initial exposure of the crew to the simulator, so that pilots' behavior and performance would reflect their actual skills in the airplane with as little contamination as possible from potential adaptation to a particular simulator configuration. The second experiment, Training and Transfer testing, examined the use of simulators as *training* tools for aviating skills, skills that would eventually need to be transferred to the airplane. That is, the experiment assessed the degree to which motion affected the training of skills and, most importantly, the transfer of those skills to the airplane. Training transfer was measured by comparing the effect of training received in the simulator, with and without motion, on performance and behavior in the simulator with motion (as a stand-in for the airplane, "quasi-transfer" design).

Two test maneuvers (i.e., pilot tasks) were chosen to maximize satisfaction of criteria described in the literature as diagnostic for the detection of a motion requirement, given the constraint that the experiment was conducted in the context of an FAA approved training program. These criteria included 1) closed loop, to allow for motion to be part of the control feedback loop to the pilot; 2) unpredictable and asymmetric disturbance, to highlight an early alerting function of motion;[8] 3) high gain and high thrust, to magnify any motion effects; 4) high workload with crosswind and low visibility, to increase the need for redundant cues such as provided by motion, out-the-window view, instruments and sound; and 5) short duration, to prevent pilots from adjusting to a lack of cues. Engine failures on take-off with either rejected take-off (RTO) or continued take-off (V_1 cut) were deemed fulfilling most of these criteria, while requiring minimum disruption to the host airline's existing training program. To prevent bias, the state of the motion system was kept concealed from all participants.

A laptop computer was programmed to control the simulator and record events with minimal I/E intervention, eliminating the need for the presence of an experimenter that might have contaminated the regular training/evaluation environment. It also enabled the I/E to focus on behavior and performance of the crew. Most importantly, it also eliminated any need to inform the

I/E (or the crew) of the interest in motion and the motion state of the simulator for each maneuver, thus minimizing any bias. To prevent the pilots from guessing which maneuvers were to come during the final testing, they were given two normal take-offs without being informed about the lack of engine failures, with the motion platform still in its original configuration.

The chronology of an experimental session is explained next. First, the crews did one V_1 cut followed by one RTO (First Look evaluation). Half of them did it with the motion system on (Motion group) and the other half with the motion system off (No-Motion group). Any additional training needed to reach the company standards for RTO and V_1 cut came next, with motion on or off depending on group. At most, there were two additional training trials for each type of maneuver. After Training, all participants filled out a questionnaire. This was followed by two normal take-offs with the same motion configuration. Then the crews did one last V_1 cut followed by one last RTO with motion on for all crews (Transfer). After Transfer testing, all participants filled out a second questionnaire, to see whether their opinions had changed after all had experienced motion.

The stimulation of the PF by the simulator and the pilots' responses were measured by recording 78 simulator state and control input variables at a high sampling rate, resulting in a vast amount of objective data on simulator performance and pilot performance and behavior/workload. Two forms of subjective data were also collected. First, at the conclusion of each maneuver the I/E provided a grade for the just-completed maneuver. Second, as already mentioned in the previous paragraph, at the end of the training period and again at the end of the transfer period all participants were queried on PF performance and workload as well as simulator comfort and acceptability. From this set of data, four types of results were obtained: 1) the motion stimulation at the PF station, 2) the effect of motion on the performance of the PFs as perceived by the I/Es and reflected in their grading, 3) the relationship between I/E grades and the objective measures of pilot performance/workload and whether this relationship was affected by motion; and 4) the effect of motion on measured performance and workload of the PFs, and the effect of motion on participants opinion regarding PF performance /workload and simulator comfort and acceptability.

MOTION STIMULATION PROVIDED BY THE TEST SIMULATOR

For the test simulator, the actually measured roll and longitudinal accelerations followed the airplane model fairly well given the limitations inherent to all simulators. For vertical acceleration, however, the motion system of the test simulator did not respond much to the command provided by the equations of motion. This is especially true for V_1 cut maneuvers. However, because the engine failures used in our experiment do not produce much vertical acceleration, the lack of vertical acceleration cuing may not be very important.

More important, however, is the finding that failure-induced lateral acceleration was not well represented by the motion system of the test simulator. Not only was it greatly attenuated, but visual inspection of the measured response does not lead to an easy distinction of failure-induced lateral acceleration, unlike the response derived from the equations of motion (relatively high peak shortly after engine failure).[7] This may represent a significant deficiency in pilot stimulation, because lateral acceleration may act as a useful cue for proper failure recognition and for initiation of appropriate response. Further research, however, still need to be done to examine the importance of lateral versus other cues in failure recognition.

I/E GRADES

The grade distribution obtained by the two groups at First Look evaluation and Transfer is depicted in Fig. 1. The possible grades were 1 (unsatisfactory), 2 (FAA Practical Test Standards (PTS)[9]), 3 (company standards), and 4 (excellent). The experimental sessions appeared to have been effective in simulating a real training session in that the crews' performance improved across the session. Specifically, combining the two motion groups (or looking at them individually), the grades for RTOs and V_1 cuts improved across the training trials. This was even stronger for the V_1 cuts, which elicited lower grades than the RTOs during First Look, but caught up by Transfer.

Platform motion had no effect on the grades that were provided by the I/Es at any time for either the RTO or the V_1 cut or for the normal take-offs. That is, platform motion did not affect First Look evaluation in the simulator, nor did it affect the grades at Transfer to the simulator with motion. The latter was true when comparing the group means and the number of low vs.

high grades in each group (i.e., grades of 3 and 4 vs. grades of 1 and 2). However, on V_1 cut at Transfer the crews which previously had motion did receive more grades of 2 than the crews who had not previously had motion, and fewer grades of 1 (none actually). Despite this single effect of motion, there was no effect of motion on the course of Training or on the amount of Training required before reaching the criterion needed to move onto Transfer for either of the maneuvers. For the complete statistical analyses, see Ref. 7.

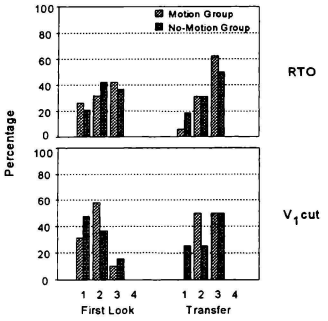


Fig. 1 Grade Distribution

MEASURED PERFORMANCE AND WORKLOAD OF PILOT FLYING

From the 78 variables recorded in the experiment, a set of criterion measures was derived for determining whether or not motion had an effect on training and evaluation of the tested pilot task. These were categorized into performance and workload/behavior measures. Performance measures reflect a pilot's control precision and efficiency in handling the airplane by measurements such as flight path deviations and reaction time. Workload/behavior measures describe how a pilot uses the controls by measurement of control inputs. A guide to the determination of the measures was provided by the PTS and by the company standards of the host airline itself. An additional goal was to capture performance and workload immediately after the engine failure, because disturbance motion was expected to act as an alerting cue to the pilots that would enhance early performance. The list of the measures can be seen in Ref. 7. Most of the measures were computed over the 15 second time period following an engine failure. Exceptions include measures of reaction times and Time to Reach 400 ft Altitude. In general, lower numerical values of the measures indicate better performance or lower workload.

The effect of motion on First Look evaluation, Transfer of training to the simulator, Training progress, and improvement from last training trial to Transfer testing was examined. Because the I/Es shared the motion platform with the pilots, and thus might have been affected by the motion status of the simulator in their grading criteria, the relationship between I/E grades and the objective measures was examined by performing regression analyses. In addition to determining whether the presence or absence of motion influenced which measures I/Es considered for grading, these analyses helped to determine criterion measures.

In this paper, only measures that are either listed in the PTS, were used by the instructors for grading, or showed an effect of motion are discussed. See Ref. 7 for a full report on all the analyses. For each measure, the statistical power was determined (i.e., the smallest effect that could be detected given the idiosyncratic variability between crews with a probability of .80). The power of the experiment was found to be sufficient to capture any operationally relevant effects.

Relationship Between Objective Measures and I/E Grades

Linear and logistic regression analyses on the relationship between the grades and the objective measures were used to infer the I/Es' grading criteria and whether the platform motion had an affect on these criteria. Although the logistic regression was considered to be more appropriate for cases involving ordinal data (like the grading system used here), the results of both regression analyses were quite similar. The regression models obtained were not meant to model I/E's decision process in determining the grades, which is actually very complex. They were only used to examine whether any available measures contributed to the I/E's grading criteria.

For RTOs, regardless of whether the platform motion was on or off, the measures of lateral and heading deviations played an important role in predicting I/E grades. For V_1 cuts, the results of the regression analyses suggest that the platform motion status may affect grading. In both motion-on and motion-off conditions, lateral measures seemed to affect I/E grades. However, the level of importance of other types of measures in the I/Es' grading criteria depended on the status of the platform motion. Notably,

longitudinal measures appeared to matter mainly when the platform motion was on.

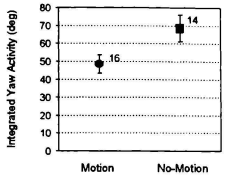
Given that I/Es may have used different grading criteria dependent on motion status, the effect of motion on grades before Transfer testing may have been masked. This appears to be a possibility at least for V_1 cuts, where the No-Motion pilots would have been able to get away with worse performance on longitudinal measures. Later findings from the objective data analysis, however, showed that the differences in the longitudinal performance between the two groups were negligible. Moreover, the regression models obtained accounted for only a small portion of the variance in the grades. These findings render the possibility that differences between the grades assigned to the two groups were masked unlikely.

First Look Evaluation, RTOs

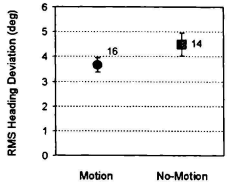
As shown in Fig. 2,[¶] the presence of motion significantly improved yaw performance of the pilots (indicated by Integrated Yaw Activity, which is the integral of the absolute yaw rate for 15 seconds after engine failure). No effects of motion, however, were found on any other performance or workload measures, including performance in heading and lateral deviations (Fig. 2), which strongly related to I/E grades. This indicates that the presence of motion did not affect First Look evaluation of RTOs in any operationally relevant manner.

First Look Evaluation, V_1 Cuts

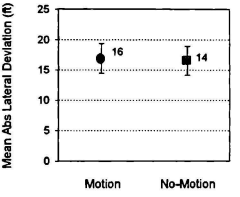
No statistically significant differences for either performance or workload measures were found between groups as a function of motion for First Look evaluation of V_1 cuts, although the Motion group was found to control pitch angle marginally more steadily than the No-Motion group ($p<.1$) (Fig. 3). Physically, however, this difference was less than one degree in average STD. Moreover, this slight advantage in pitch angle control was not accompanied by improvements in any of the other longitudinal performance measures. This, together with the fact that there was practically no simple correlation between STD Pitch Angle of Motion pilots and grades ($r^2=.01$), and even the stepwise regression model selecting three more longitudinal measures accounts for no more than 30 percent of the variance in the grades, suggests that the platform motion would not affect what grades PFs achieve during First Look evaluation. This result also validates the subjective grade results presented earlier.



$p=.033$

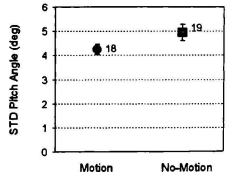


$p=.126$



$p=.906$

Fig 2 RTO First Look: Directional Performance



$p=.096$

Fig. 3 V_1 Cut First Look STD Pitch Angle

[¶] In this and subsequent figures, numbers next to data points refer to sample size.

Training Transfer, RTOs

Training Transfer was tested for all crews on the simulator with motion activated as a stand-in for the airplane. Despite the fact that the Motion crews were trained and tested on the same simulator configuration, they did not do any better than the No-Motion crews with any RTO performance and workload measure. Additionally, the power of the experiment was generally higher after training, and still no effects of prior motion were found. One *caveat* is that for heading control, although there was no difference between the two groups, more No-Motion crews improved than Motion crews between the last training and the Transfer testing (Fig. 4). This may indicate an effect of adding motion, although during Transfer testing the two groups performed at the same level (as just described).

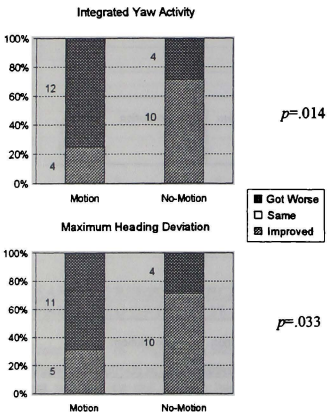


Fig. 4 RTO Last Training vs. Transfer: Directional Control Performance

Training Transfer, V_1 Cuts

In terms of performance, the most notable differences between the two groups were on Integrated Airspeed Exceedance (the integral of the absolute airspeed deviation outside the (0,+5 knots) band from the recommended V_2) and STD Pitch Angle. The Motion group controlled airspeed better (p=.006) at the expense of increased STD Pitch Angle (p=.025) (Fig. 5). Physically this can be interpreted as the Motion group controlling airspeed more successfully by adjusting pitch angle more aggressively than the No-Motion group. Note that speed control is critical in V_1 cuts, because it involves safety (e.g. for clearing obstacle and maintaining a margin above stall speed). The Motion group also displayed higher Integrated Yaw Activity compared to the No-Motion group (p=.024) (Fig. 6). However, this did not appear to result in any differences in heading control or other directional performance measures. No other statistically significant performance differences were found.

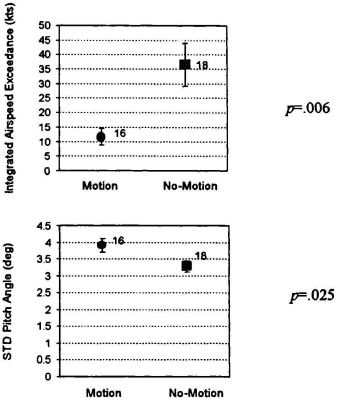
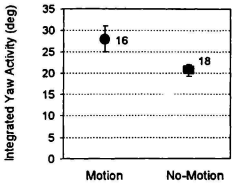


Fig. 5 V_1 Cut Transfer: Longitudinal Performance

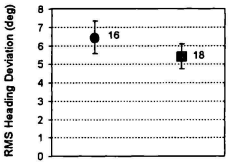
With regard to workload during V_1 cuts, the Motion group had fewer wheel reversals than the No-Motion group (p=.059), whereas the No-Motion group had fewer pedal reversals than the Motion group (p=.008) (Fig. 7). The increased number of Wheel Reversals of the No-Motion group was not accompanied by any lateral performance differences. The increased number of pedal reversals of the Motion group, however, was accompanied by an increase in Integrated Yaw Activity, as was discussed earlier. The difference was not apparent at First Look, nor did a combined Analysis of Variance (ANOVA) of Motion/No-Motion by First Look vs. Transfer find a significant interaction, probably due to the variability in number of pedal reversals for the Motion group during First Look. The questionnaire data indicated that the Motion group felt the pedal was less like the airplane than the No-Motion group did.

Although a few statistically significant differences between the groups trained with and

without motion were found during V_1 cut Transfer testing, the size of these differences on average were only about 1.5 knots exceedance per second for airspeed, half a degree RMS for pitch angle deviation, and half a degree per second for yaw rate. Such differences are very small compared to about 110 knots desired nominal airspeed and about 10 degrees nominal pitch angle during climb, and may be considered operationally irrelevant.



p=.024



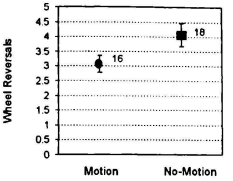
p=.354

Fig. 6 V_1 Cut Transfer: Directional Performance

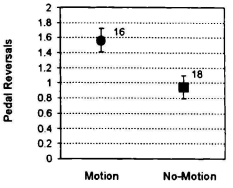
Training Progress, RTOs

No statistically significant differences in improvement from first to last training trial were found between groups for any of the measures (all p>.2). This suggests that the platform motion did not affect the training progress of the pilots.

Also, the overall number of crews (Motion and No-Motion) improving in lateral performance and workload measures was significant for most measures, with the exception of Integrated Yaw Activity with no overall improvement and pedal reversals, which actually increased after training. When looking at the groups separately for these two measures, neither of the groups shows any improvement or deterioration. This confirms that the pilots generally did improve during training regardless of the motion status of the simulator.



p=.059



p=.008

Fig. 7 V_1 Cut Transfer: Wheel and Pedal Reversals

Training Progress, V_1 Cuts

The course of training for V_1 cuts reflected the Transfer results. For longitudinal control during V_1 cuts, motion improved Training progress for speed control (Integrated Airspeed Exceedance), but at the cost of pitch angle control (STD Pitch Angle) (Fig. 8). Progress in directional control (i.e., RMS Heading Deviation, Integrated Heading Exceedance, and Maximum Heading Deviation) was also negatively affected by the presence of motion during Training (p<.1). The Training progress on lateral control was not affected by the presence or absence of motion. Also, there was no difference for workload between the two motion groups.

The data indicate that the No-Motion group improved on more measures than the Motion group. While Motion crews improved in Integrated Airspeed Exceedance and STD Column Position only, the No-Motion crews improved in Integrated Bank Angle Exceedance, Heading Deviation, Time to Reach 400 ft Altitude, and STD Pitch Angle. During Transfer, however, the No-Motion group surpassed the Motion group only with steadier pitch angle and yaw activity; and the actual size of these differences was very small.

The above discussion indicates that the training without motion was at least as effective as the training with motion, and the earlier results on Transfer show that although some differences were found in

training progress between the two groups, they did not translate into operationally relevant differences during Transfer.

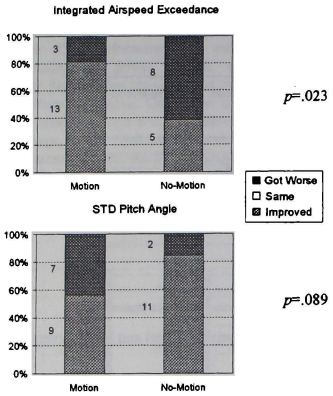


Fig. 8 V_1 Cut First vs. Last Training: Longitudinal Performance

QUESTIONNAIRE DATA

Each of the PFs and PNFs was given two questionnaires (i.e., one after Training and one after Transfer) that each had six questions (i.e., control precision, workload, gaining proficiency, simulator comfort and acceptability). Each I/E was also given two questionnaires, each with five questions (i.e., the same questions as above, but without acceptability). PFs responded always with reference to themselves. PNFs and I/Es referred to the PFs, with the exception of comfort and, for the PNF, acceptability.

Despite all of these questions, only four differences were found between the Motion and No-Motion crews. 1) After Training, the PNFs from the No-Motion crews rated the control precision of the PFs better than the PNFs from the Motion crews did. 2) The PFs from the No-Motion crews, once transferred to the simulator with motion, rated their control precision higher than their motion-trained counterparts. This is possibly because of the contrast between the added motion and the lack of motion they had been experiencing. 3) In contrast, after Transfer, the I/Es gave higher ratings for performance to the PFs from the Motion group than to the PFs from the No-Motion group. 4) Looking across both questionnaires, the PFs from the No-Motion crews gave better ratings to the simulator for training ("gaining proficiency") than the PFs from the Motion crews.

All together the subjective responses of the pilots and the I/Es did not indicate that the motion used in this study had any impact on the PFs' performance. It also had very minimal impact on the pilots' perception of their own performance, workload, ability to gain proficiency, comfort, or their acceptability of the simulator.

CONCLUSIONS AND RECOMMENDATIONS

The results of this study indicate that the motion provided by the test simulator, which may or may not be typical of other FAA qualified Level C flight simulators, does not, in an operationally significant way for the tasks tested, affect either First Look evaluation, Training progress, or Transfer of training acquired in the simulator with or without motion to the simulator with motion. It also doesn't consistently affect the PFs', PNFs', and I/Es' subjective perception of the PFs' performance, workload, and training, or of their own comfort in the simulator. Neither does it affect the acceptability of the simulator to the PF and the PNF.

Two *caveats* have to be kept in mind, however. First, the simulator used in this study may not have provided sufficient motion stimulation to be effective. The measurements indicate that the simulator may have failed to provide lateral acceleration cuing representative of the aircraft for the test maneuvers (RTO and V_1 cut).

A second *caveat* is that the current study used the simulator with motion as a stand-in for the airplane. Although some may believe that this quasi-transfer design needs to be validated, others say that high-level simulators have been validated as a stand-in for the airplane by many years of use of the simulator for total flight training. Also, given that the motion-trained group transferred to the same simulator configuration that they had been trained in, whereas the No-Motion group transferred to a configuration that was new to them (i.e., the motion configuration), the Motion group should have had an advantage. Based on the quasi-transfer results, it is unlikely that it would have had a greater advantage transferring to an airplane.

Clearly additional steps must be taken to determine the extent to which it may or may not be appropriate to draw generalizations from these results. These should include a comparison of the objective

measures from the motion system used in this experiment with such measures taken from other FAA qualified Level C simulators to determine whether or not the motion used in the present study is representative. This should be followed by an investigation on whether operationally relevant effects of motion would be found with a simulator where the motion is manipulated to assure that it is representative of the airplane for the maneuvers selected. Additional maneuvers that may be diagnostic and a different pilot population should be tested as well. Ideally, some validation of the quasi-transfer design with a real airplane would also be undertaken.

ACKNOWLEDGMENTS

This work was funded by the Office of the Chief Scientist for Human Factors of the Federal Aviation Administration, AAR-100. The FAA Program Manager was Dr. Eleana Edens. The manager of the FAA Advanced Qualification Program Dr. Thomas Longridge identified the need for this work. We thank them for their guidance throughout the project.
The opinions expressed are those of the authors and not necessarily those of the Department of Transportation, the Federal Aviation Administration, or the U.S. Government.

REFERENCES

1. Bürki-Cohen, J., Boothe, E.M., Soja, N.N., DiSario, R.D., Go, T., and Longridge, T., "Simulator Fidelity—The Effect of Platform Motion," proceedings of the Royal Aeronautical Society conference on Flight Simulation—The Next Decade, London, May 2000.
2. Longridge, T., Ray, P., Boothe, E.M., and Bürki-Cohen, J., "Initiative Towards More Affordable Flight Simulators for U.S. Commuter Airline Training," paper presented at the Royal Aeronautical Society Conference on Training—Lowering the Cost, Maintaining the Fidelity, London, May 1996.
3. Transcript# of the Joint FAA/Industry Symposium on Level B Airplane Simulator Aeromodel Validation Requirements, Washington Dulles Airport Hilton, March 13-14, 1996.
4. Transcript# of the Joint FAA/Industry Symposium on Level B Airplane Simulator Motion Requirements, Washington Dulles Airport Hilton, June 19-20, 1996.
5. Bürki-Cohen, J., Soja, N., and Longridge, T., "Simulator Fidelity Requirements: The Case of Platform Motion," paper presented at the International Training and Education Conference and Exhibition, Lausanne, Switzerland, 1998.
6. Bürki-Cohen, J., Soja, N., and Longridge, T., "Simulator Platform Motion—The Need Revisited," The International Journal of Aviation Psychology, Vol. 8, No. 3, 1998, pp. 293-317.
7. Bürki-Cohen, J., Soja, N.N., Go, T.H., Boothe, E.M., DiSario, R., and Jo, Y.J., "Simulator Fidelity: The Effect of Platform Motion," Report No. DOT/FAA/RD-00/XX, in preparation.
8. Gundry, J., "Man and Motion Cues," paper presented at the Third Flight Simulation Symposium, London, April, 1976.
9. Federal Aviation Administration, "Airline Transport Pilot and Type Rating Practical Test Standards," FAA-S-8081-5B, U.S. Government Printing Office, Washington, D.C., July 1995.

Available in electronic format from Dr. Thomas Longridge, Advanced Qualification Program Manager, AFS-230, tel. (703) 661-0275

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37257

AIAA-2000-4297

MOVE OR NOT TO MOVE? CONTINUOUS QUESTION

Cezary Szczepanski, Ph.D., M.Sc.Eng, Warsaw University of Technology, ETC-PZL Aerospace Industries, Poland
Dick Leland, Environmental Tectonics Corporation, Southampton, PA

ABSTRACT

The authors' many years experience in design and manufacture of flight simulators, aviation training devices, and ground vehicle simulators have provided extensive understanding regarding the importance of motion systems in simulators. This paper presents our conclusions on the necessity of motion systems in flight training simulators for fixed wing aircraft and helicopters used by the armed forces. Motion system options considered were no motion, less than 6 DoF, and 6 DoF. In preparing the paper, the following factors were considered:

- Type of Simulated Aircraft,
- Range of Simulator Usage,
- User of The Simulator,
- Technological Aspects,
- Pilots' Psychological and Physiological needs,
- Financial Aspects.

Our conclusion is that motion systems are necessary for flight simulation, especially when training flight tasks where motion stimuli are important for the pilot to determine appropriate control inputs. When the simulation only stimulates the pilot visually, there is a great deal of missing information that is normally present in flight and learning transfer is often adversely affected. Pilots should be trained in an appropriate motion system equipped simulator in order not to sacrifice training effectiveness. This is especially essential when pilots are trained for high G tolerance and Spatial Disorientation avoidance.

INTRODUCTION

The term "flight simulator" connotes a device that imitates all aspects of flight in the aircraft. This intuitive "definition" generally reflects its main goal. A high fidelity imitation means a very precise creation of visual, motion, sound, smell, tactile and taste cues that can be perceived by the pilot. Total fidelity simulation would stimulate all the human senses with the proper input signals.

Copyright © 2000 by Environmental Tectonics Corporation. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

It is practical to minimise the number of cues and their complication thereby minimising the cost of simulation while still maintaining an acceptable level of similarity to the real flight. For example, it is not always necessary or practical to reproduce taste or smell cues because most often they are not important for the pilot. Clearly visual stimulation is necessary for simulation. In fact, some people feel that with adequate visual stimulation, motion cueing is unnecessary. This argument has persisted since the early 1970's and has been more an issue of economics than training quality. Decisions made in the 1970's on the effectiveness and role of motion in simulation were also based on the limited motion base technology and limited understanding of the role of the Human perceptual system at that time. Accordingly, the role of the motion stimuli during the flight simulation is worth further consideration.

The analysis presented in this paper was based on a review of an extensive military aircraft simulator database[1]. A total of 390 simulators of various configurations were considered. Of the 390 simulators considered, 290 were fixed wing aeroplane simulators and 100 were helicopter simulators. The database did not count multiple copies of the same simulator configuration as separate simulators. Rather, each simulator was at least one device ordered by a separate user for a certain version of aeroplane. Accordingly, the actual total number of simulators produced and used is most certainly much bigger (e.g. eight Flight Training Devices for the Hawk aeroplane were counted as a one item in our database). Also, we should mention that the analysed database contained simulators and flight training devices that were completed and in service at the end of 1998 and some from 1999. The database did not include motion based simulators used in the aviation medical field. Motion based aviation medical trainers are discussed separately at the end of this paper.

The authors acknowledge that aircraft simulators are constantly being replaced, modified, installed and taken out of service. Accordingly, not all existing aircraft simulators may have been included in our analysis. Nonetheless, the database that we analysed is significant in number of simulators and is representative of currently used simulation systems.

The type of aeroplane represented influences the requirement for motion system capability. For analysis purposes, it is convenient to divide the aeroplane simulators and flight training devices into the two basic groups:

- Fixed Wing (FW) Aeroplane
- Rotary Wing (RW) Aircraft.

The FW Aircraft group can be further divided into the following types:

- Fighter and Fighter-Bomber Aircraft
- Vertical/Short /Takeoff and Landing (V/STOL) Aircraft
- Reconnaissance Aircraft
- Trainer Aircraft
- Attack Aircraft
- Heavy Bomber and Transport Aircraft
- Tanker and EW Aircraft

Devices used in pilot training are broadly categorised as either Flight Simulators or Flight Training Devices. Generally, the designation of Flight Simulator has connoted a higher level of fidelity and full flight training function while the designation of Flight Training Device has connoted a lower level of fidelity and/or more specialised training functions. However, considerable overlap exists within these broad categories and this imprecise labelling has created considerable confusion.

The range of simulator uses reflects the type and level of pilot and crew training performed on the simulator. Also, it presents the user interest in simulator characteristics. Often these characteristics are reflected in the name used for the device.

The following names are commonly used:

FLIGHT SIMULATORS

- FFS – Full Flight Simulator,
- FMS – Full Mission Simulator,
- FMT – Full Mission Trainer,
- WSS – Weapon System Simulator,
- WDT – Weapon Delivery Trainer,
- FBS – Fixed Base Simulator,
- FNPT–Flight and Navigation Procedures Trainer
- FTD – Flight Training Device,

FLIGHT TRAINING DEVICES

- IFT – Instrument Flight Trainer,
- CSS – Cockpit Systems Simulator,
- CPT – Cockpit Procedures Trainer,
- UTD – Unit Level Training Device,
- RCT – Rear Crew Trainer,
- WCT – Whole Crew Trainer.

In contrast to civil aviation simulators, little nomenclature standardisation exists for military flight simulators and training devices. Accordingly, many different names have been used for apparently similar simulator devices (e.g. FMS or FMT and WSS or WTD are often used for the same types of devices. On the other hand, sometimes a WTD (depending on its description) is equivalent to a FMT. A similar situation exists in the flight training device class. Often the classifications of FTD, IFS, CSS, and UTD, refer to similar levels of complexity and function.

While the plethora of names to describe similar capability can be frustrating, one should remember that the above listed devices were produced over a period of many years when the different names were used. Since formal classification standardisation did not exist, names often reflected the up-to-date technologies and they were used as marketing tools.

The ultimate user of the simulator is very often not clearly defined. However, we can estimate types of users by examining the range of simulator usage. User type is an important consideration in analysing motion system requirements.

Technological aspects should also be taken into consideration. Historically, technology possessed by the simulator producer has been a primary reason for the type of motion system incorporated. A primary factor that has historically been significant in the motion versus no motion debate stemmed from these limitations in motion technology and motion control technology that resulted in incorrect motion cueing. As one U. S. Army helicopter pilot remarked to one of the authors "No motion is better than bad motion."

Financial aspects are another important consideration for motion system integration. In fact, a primary factor in the motion system versus no motion system debate that started in the 1970's was the high cost of motion platforms. Financial aspects continue to be of fundamental importance for both the producers and users of simulators.

To aid our motion system analysis, we established a general division into the three groups of pilot training devices with regard to motion capability. The first group includes devices without motion systems. The second group includes devices with motion systems that have less than six degrees of freedom (example shown in Figure 1). The third group includes devices with 6 degrees of freedom motion systems (example shown in figure 2). The second group was created because of existence of a number of pilot training devices that employ between one and four axes or degrees of freedom motion systems. These trainers fall into two subcategories:

- Trainers designed for a specialised training tasks (e.g. Spatial Disorientation training and High G training)
- Low cost solutions for Flight Training Device motion cueing.

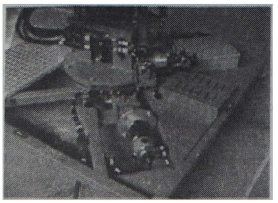


Figure 1: 3 Axes Electromechanical Motion System

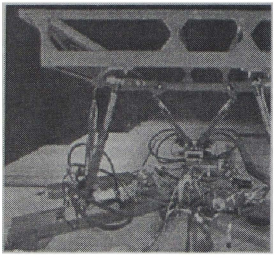


Figure 2: 6 DoF Motion System

The objectives of this research paper were to analyse and decide on the importance of motion stimulation for flight simulation. To move or not to move – let us start to find the final conclusion.

1. TYPES OF SIMULATED AIRCRAFT

This is the most important criterion. It reflects the general opinion on the importance of the motion system in the flight simulation for different types of the aircraft.

1.1. Helicopters

Among 100 helicopter simulators listed in database [1] we found the following:

- Group One (no motion system): 16 devices
- Group Two (less than 6 degrees of freedom motion systems): 4 devices
- Group Three (six degrees of freedom motion systems): 80 devices

Group One, consists of 5 CPTs and 2 FMSs for the Apache WAH-64, and 1 IFT for the Bell UH-1H. The remaining 8 devices are FBSs for different types of helicopters.

Group Two consists of 2 Lynx and 2 Sea King trainers. These trainers have 3 degrees of freedom motion systems. Although these trainers have limited motion capabilities, they provided budget savings for the user.

Group Three is the largest group, includes 80 FFS, FMS, FMT and WSS helicopter simulators.

The helicopter flight dynamics characteristics generate motion stimuli that are perceived by the pilots during the real flight. Pilot control reaction from motion stimuli is from five to eight times quicker than for visual stimuli. Pilot reaction time during actual flight is often the most critical factor. Accordingly, pilots often use these stimuli as a main source of information on the flight condition or state. Accordingly, properly imitating the motion stimuli during the flight simulation is very important.

Based on empirical data, the six degrees of freedom motion system appears to have become a standard for rotary wing simulation. A possible enhancement to the traditional 6 DoF motion system would be the introduction of 360 continuous yaw cueing capability. However, this ability must be introduced in a way that allows for simultaneous motion cueing in pitch, roll, heave and 360 degrees continuous yaw. Currently, no 6 DoF motion system supports this requirement. A 4 DoF system (pitch, roll, heave, and 360 degrees continuous yaw) that will support +/- 360° continuous yaw motion simultaneously with pitch roll and heave motion cueing is currently under development (figure 3).

The 6 DoF motion system's present ability of replicating these motion stimuli is good enough for most helicopter training purposes. The 4 DoF system with continuous yaw capability will expand these training capabilities to include auto-rotation and Spatial Disorientation.

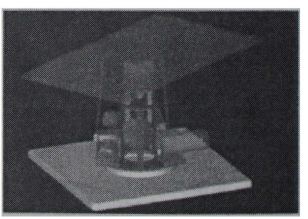


Figure 3: 4 DoF Motion System

1.2. Fixed Wing Aircraft

This group is much more diverse from the flight characteristics point of view. There are heavy, big and dynamically slow aeroplanes, lighter multiplace aircraft with similar dynamics and highly manoeuvrable, dynamically quick aircraft. Ideally, their associated flight training devices and simulators would have been appropriate to these characteristics. Our analysis of the subgroups of aircraft listed above follows.

1.2.1. Fighter and Fighter-Bomber Aircraft

The most distinguishing characteristic of these aircraft is their high degree of manoeuvrability. This subgroup of pilot trainers is the most populated; containing 140 devices. Breakdown by motion system categorisation is as follows:

- Group One (no motion system): 107 devices
- Group Two (less than 6 degrees of freedom motion systems): 5 devices
- Group Three (six degrees of freedom motion systems): 28 devices

Group One consists of FBS, UTD, CPT, RCT, WDT, FMS and FFS devices. The main differences between these devices, from technical point of view, lie in the visualisation systems. There are only 4 devices for the Fighter-Bomber aircraft and the others are for fighter and interceptor aircraft.

Group Two consists of 3 FTDs with two degrees of freedom motion systems for the F-104 Starfighter, 1 FFS with a three degrees of freedom motion system for JAS-37 Viggen, and 1 WDT with a three degrees of freedom for Tornado GR1.

Group Three consists of CPT, RCT, FFS, and FMS devices. Among them there are 20 simulators of fighter interceptors and 8 of Fighter-Bomber type aircraft. Interestingly, the Fighter-Bomber aircraft simulators were mostly done using six degrees of freedom motion systems. This may be connected with their operational flight characteristics.

1.2.2. V/STOL Aircraft (e.g. Harrier)

We decided to consider simulators for this aeroplane type as a separate group because of these aircraft's unique flight characteristics (e.g. during some flight stages one can treat it as a helicopter at the others as a fighter). Generally, this aircraft performs as a highly manoeuvrable aeroplane. We have counted the following numbers of simulators. Breakdown by motion system categorisation is as follows:

- Group One (no motion system): 5 devices
- Group Two (less than 6 degrees of freedom motion systems): 2 devices
- Group Three (six degrees of freedom motion systems): 3 devices

Group One consists of FMS and FFS devices used by the navies and marines.

Group Two consists of the FFS devices used by the Royal Air Force for Harrier GR 1/3 and Marine Corps for the AV-8A Harrier.

Group Three consists of FFS devices used by the Royal Air Force, Royal Navy and Indian Navy, the last two for the Sea Harrier.

These devices are very complicated and allow for the substantial range of training. The differences in the motion systems used may stem from the different users and different requirements for both aeroplane and simulator.

1.2.3. Reconnaissance Aircraft

There is a rather small group of simulators reflecting the small number of these highly specialised aircraft. We counted here SR-71 Blackbird and Jaguar GR1 aircraft simulators. Breakdown by motion system categorisation is as follows:

- Group One (no motion system): 0 devices
- Group Two (less than 6 degrees of freedom motion systems): 1 device
- Group Three (six degrees of freedom motion systems): 3 devices

There are no simulator or flight training devices without motion systems (Group One).

Group Two conists of 1 FFS with 3 degrees of freedom motion system for the Jaguar.

Group Three consists of 3 FMS with 6 degrees of freedom motion system for the SR-71 Blackbird. This is very possibly a result of the mission types that are performed by pilots of these aircraft. They are usually long flights (at least longer than interceptor's missions). Blackbirds used to do the cruising missions on high altitudes and Jaguars NOE or low level flights. But both of them need to have simulators with six degrees of freedom motion systems for the proper imitation of the important motion stimuli to accomplish important training activities such as emergency procedures, flight control malfunctions, mission tactics, and post stall recoveries. Reconnaissance training missions are difficult and expensive to perform in-flight. Accordingly, the economical and efficient way of training these pilots is by using sophisticated flight simulators.

1.2.4. Trainer Aircraft

Among training aircraft simulated were piston engine, turbo prop engine and jet engine aircraft. They are used for initial training, screening flights and advanced training. These areas indicate the simulator use ranges. Because of the large number of aircraft in this category (used for training of many candidate pilots) this group of devices is quite large. The total number in our database is 47. Breakdown by motion system categorisation is as follows:

- Group One (no motion system): 31 devices
- Group Two (less than 6 degrees of freedom motion systems): 6 devices
- Group Three (six degrees of freedom motion systems): 10 devices

Group One consists of IFS and IFT for TA4J and T45, also FTD and FMS for Hawk. But almost all of the training aircraft have their training devices represented here.

Group Two consists of FTD and FFS with three degrees of freedom for Hawk, and 2 FFS with two and three degrees of freedom motion system for Tucano.

Group Three consists of full flight and full mission simulators for the above mentioned aircraft, mainly for the jet trainers.

The above devices are typically used by the training centres and pilot training academies. They are used for extensive training of a large number of trainees. They are also produced in the numbers bigger than other types of simulators ordered once.

1.2.5. Attack Aircraft

There is a small group of specialised aircraft. Their pilot training requirements are mostly connected with the mission training. These requirements are reflected in the simulators' structure. Breakdown by motion system categorisation is as follows:

- Group One (no motion system): 12 devices
- Group Two (less than 6 degrees of freedom motion systems): 0 devices
- Group Three (six degrees of freedom motion systems): 3 devices

Group One consists of two types of FMS, FBS and FTD. The main purposes of these devices are mission and combat training, upgrading the crew skills in armament usage and training the crew in the common actions. These training objectives necessitate very high cockpit fidelity to support aircraft and system specific procedures training.
In this case, the motion system is not a crucial element.

Group Two had no training devices represented.
Group Three consists of only three devices. These devices are primarily used by aircraft qualification training centres. High level motion systems are incorporated into these flight simulators since they are primarily used for transient or initial pilot training. These trainees do not have an internalised model of the aircraft's flight control inputs and associated dynamic response yet.

1.2.6. Heavy Bomber and Transport Aircraft

These types of aircraft are similar in performance to civil airline aircraft. Accordingly, much of the crew training is similar to civil aviation pilot and crew training. Accordingly, the simulators used for this group of aircraft are similar to the civil aircraft simulators. Since civil aircraft simulators are more standardised, we can expect devices in this category to also be more standardised. Breakdown by motion system categorisation is as follows:

- Group One (no motion system): 7 devices
- Group Two (less than 6 degrees of freedom motion systems): 0 devices
- Group Three (six degrees of freedom motion systems): 31 devices

Group One consists of 1 WDT for B2 bomber and 6 FTD and CSS for C17A and C-130 Hercules. These training devices are used for the crew training in on-board systems operation or procedural actions. In these training activities and levels of learning, motion stimuli do not play a main role.

Group Two had no training devices represented.

Group Three consists of FFS for the B-2, C-17A, and C-130 aircraft. These simulators are compliant with the technical characteristics described in the FAA and JAA flight simulator standards, where the six degrees of freedom motion system is obligatory for the pilot and crew training.

1.2.7. Tanker and EW Aircraft

From the flight dynamics characteristic point of view tankers and Electronic Warfare aircraft are very similar to the Heavy Bomber and Transport group and, in turn, to the civil airliners. Accordingly, we can expect these simulators to be used for the similar training purposes.

Breakdown by motion system categorisation is as follows:

- Group One (no motion system): 2 devices
- Group Two (less than 6 degrees of freedom motion systems): 0 devices
- Group Three (six degrees of freedom motion systems): 28 devices

Group One consists of RCT for Nimrod and E2.
Group Two had no training devices represented.
Group Three consists of FFS and FMS for above and all other types of aircraft.

Explanation for the above numbers is similar to the previous subparagraph. Dynamically slow, heavy, multi-crew aircraft, like airliners, need to have such simulators for the training of their crews. Realism is an important characteristic since simulator training is an integral part of airline pilots' training since little training is done in the actual aircraft for economic reasons. Accordingly, the current, standard devices for the civil pilot training are Full Flight Simulators level C to D, which are obligatory equipped with the six degrees of freedom motion systems (example shown in figure 4).



Figure 4 Level D Simulator (Military Aircraft)

Figure 4: Level D Simulator (Military Aircraft)

The 6 DoF motion system can accurately replicate most of the motion cues encountered in the normal flight regime. Limitations are encountered when attempts are made to use these types of simulators for training activities that exceed the normal flight regime. These manoeuvres include spatial disorientation episodes, post-stall gyrations, jet upset episodes, and certain control malfunctions. Replicating these flight activities is an area that is currently receiving considerable scrutiny. One requirement that seems to have emerged is the need for continuous motion cueing and G force (albeit at low G levels) generation. This may indicate the need for a specialised motion system. Examples of these systems are discussed later in this paper.

Mission training, particularly for the tankers, needs full motion stimuli to be replicated as accurately as possible. That is why we have the majority of the six degrees of freedom motion systems implemented into the training devices used for these types of aircraft.

2. USERS AND RANGE OF SIMULATOR USE

As we have mentioned in the previous paragraphs, there is little precise information regarding simulator use or training device users. Typically, we know only the owner name rather than the real user (e.g. U.S. Air Force, Navy, Army, Polish Air Force, etc.) Sometimes information on end user is also known. It helps then to analyse the device's usage range.

Simulator data analysis suggests that motion cueing is necessary when training ab initio pilots or pilots who have limited or no experience in the particular flying task that is being trained. In many flight manoeuvres, the pilot's control inputs are based on their perception of acceleration, velocity or physical position (e.g. bank angle or pitch angle), These inputs are generated during actual flight and form important inputs when the pilot interprets aircraft performance and determines proper flight control responses and inputs.

Huge amounts of information are continuously sensed by the visual system, the vestibular system, the somatosensory system (sometimes referred to as "seat of the pants"), and the auditory system and then processed by the brain. The pilot's knowledge of where he is and what the aircraft is doing as well as it's position relative to the earth's surface - his spatial orientation and situational awareness - comes from perceived elements of this information that are processed and interpreted by both the conscious and the preconscious areas of our brain.

An inexperienced pilot who is trained in a device that only stimulates the visual and auditory systems is only required to process part of the information that he will be required to process and interpret when executing the same manoeuvres in the actual aircraft in flight. Consequently, when the pilot first attempts what he has learned in a non-motion device in the aircraft, his performance typically suffers dramatically since his conscious and preconscious brain now has a much higher information processing load. The previously learned manoeuvres must then be relearned in the "new" aircraft in-flight environment, so training efficiency suffers

On the other hand, simulator users who train experienced pilots with deeply internalised models of flight control inputs versus expected aircraft response may not always need motion cueing. The exception to this is if the simulated aircraft's flight dynamics characteristics do require it (e.g. helicopter, high manoeuvrability or V/STOL aircraft).

In summary, the requirement for motion stimuli is connected with the physiology of pilot's perception and reaction. When motion cueing is the primary source of information for flight simulation, it should be included in the simulation system (e.g. instrument training for new pilots, jet upset training, post stall gyration training, in-flight manoeuvres training, and in-flight emergency procedures training). However, if motion cueing is of secondary or less importance, it does not need to be simulated (e.g. instrument flying for experienced pilots, cockpit task training, cockpit procedures training, ground emergency procedures training).

Discussions often occur when deciding on simulating motion stimuli, which are normally of secondary importance but during some states of flight becomes primary. The best solution for this dilemma is the following. If we want to train flight activities in the simulation system where the pilot normally bases decisions and control inputs on motion cueing, or when training pilots who do not have internalised flight control models and aircraft performance models, we should provide proper motion stimulation. This decision has to be determined by the end user together with the supplier of the simulation system. Otherwise, we can easily get a simulator, which will not be used in the required range or could create improper pilot's habits. .

Simulator flight training typically involves a mixture of training activities, some that require motion cueing and some that do not require motion cueing. Accordingly, training centres using simulation systems generally prefer devices that cover this broad range of usage. This approach gives them flexibility of training and the ability to easily adapt to changing most new training requirements.

Simulators used by the military flying squadrons are more often used for upgrading the pilot proficiency. The experienced pilot does not need basic flight control and or basic instrument and navigation training. Rather, the experienced military pilot's training requirements involve executing complex manoeuvres in difficult flight conditions, emergency procedures training, or new on board equipment familiarisation. For many emergency procedures and for new equipment familiarisation, the most suitable are devices without a motion system. The motion stimuli then are of less importance and as such need not to be incorporated into the device. However, for practising complex manoeuvres, motion cueing becomes very important.

Precise information on end users of systems would have aided our analysis. The former and present status of end users would also be important. To further compound the issue, some devices introduced into the service many years ago were upgraded and others were moved to different end users. However, these limitations did not change our analysis results or adversely influence our conclusions.

3. TECHNOLOGICAL AND FINANCIAL ASPECTS

The decision for or against a motion system should ideally be made by the end user and based on training objectives. However, in some cases we could suspect that a motion system was optional for the device, or it was eliminated because of financial restrictions. This situation could arise if the producer of the device offered both fixed base and a motion system versions.

Regarding devices with motion systems that featured less than 6 degrees of freedom, two situations could have existed. In the first case, the device may have been designed for a special application such as Spatial Disorientation or High G training. The second (and somewhat troubling) situation would have resulted with the acquisition of simpler devices featuring motion systems that were not fully adequate to meet the training requirements of the end user. However, one should not automatically assume that a 3 or 4 DOF system is necessarily of lower capability or lower in cost than a 6 DOF system.

Users continually place demands on producers to produce simple and economical simulators and flight training devices. Previously used technology limitations did not allow for the faithful motion simulation while resulting in complicated and expensive devices that had high operation and maintenance costs. However, with current technologies, there are many opportunities for very good simulator development at comparatively lower prices. If high fidelity (FAA Level C or D) stimulation and high payload weights are necessary, as in the case of multi-crew flight simulators, the 6 DoF motion system remains the most practical (albeit somewhat expensive) choice.
On the other hand 3 axes (pitch, roll, and yaw) or 4 DoF (pitch, roll, yaw, and heave) electromechanical motion systems can offer an economical and in some applications more effective alternative. The best, general rule of thumb is that decreasing the cost of training is important, but below a certain level one cannot expect a good, trained pilot at the end of a process.

Almost all simulator producers claim that they offer COTS devices or at least COTS elements in them. How can we say that it is a COTS flight simulator, when it is tailored according to the individual client requirements? More probable it uses COTS elements.

End user satisfaction ratings can be somewhat skewed. If, for example, after completing the

acquisition of the device and the initial period of its utilisation, the end user may find that the device does not meet the training needs. This presents a very difficult situation because the procurement has been made. Accordingly, the end users fear of being accused of purchasing the wrong device may drive a positive opinion on a device that actually has limited training capabilities. We cannot eliminate that kind of misleading information but we think that there are a few of these cases. They eventually should not affect the final analysis results.

4. DISORIENTATION TRAINING DEVICES

SD and SA are two interrelated areas where the benefits of motion based training devices are readily appreciated. Sophistication of these devices has evolved from the Barany Chair (where the air crew is just a passive rider) to fully interactive, 4 axes motion system devices.

Spatial orientation is a key component of Situational Awareness and when the former is lost, the latter often begins to break down. Accordingly, the goal of SD training is to enable the pilot to readily convert unrecognized SD into recognized SD. Pilot recognition of some illusions can be taught using the Barany chair although its capability is limited to demonstrating leans, coriollis, nystagmus and (to a degree) the effect of expectancy on resolving SD episodes. A higher level of learning occurs when the pilot not only recognizes the SD episode, but also resolves or manages it and subsequently flies through the episode, thereby developing both recognition and coping skills

Simultaneous and continuous motion cueing are essential for Spatial Disorientation training. Without such cueing vestibular illusion training is not possible and visual illusion trianing is degraded. This type of motion cueing requires a specialized motion system, that is specifically designed to support Spatial Disorientaiton training. Accordingly, a traditional 6 DoF motion system is not appropriate for this type of training.

Several motion system solutions exist. These can be divided into the following groups:

- Group One: Four axes electromechanical motion system that provides +/- 360° continuous motion in pitch, roll, yaw, and generates G forces up to 3 Gs through the use of a planetary drive system.
- Group Two: 4 Degrees of Freedom electromechanical motion system that provides +/-20° pitch, +/-30° roll, +/- 360° continuous yaw, and +/- 4 inches heave
- Group Three: 3 axes electromechanical motion system that provides +/-15° pitch, +/- 25° roll/ +/- 360° continuous yaw.
- Group Four: Single axis motion systems that provide 360° continuous motion in yaw
  Group Five: 6 DoF hydraulic motion system with a hydraulic yaw drive located on top of the 6 DoF system that adds 360° yaw.

Group One consists of 10 Spatial Disorientation training devices. These trainers include both specific aircraft cockpits and generic aircraft cockpits. The 4 axes motion system represents the latest technology in motion cueing for spatial disorientation training, jet upset training, and high performance aircraft simulation. An example of this type of trainer is the GYROLAB GL-2000 by Environmental Tectonics Corporation (ETC), shown in figure 5. The wide field of view virtual image display, and unique motion base technology combine to reproduce many of the conditions which can lead to disorientation and/or loss of situational awareness.

Additionally, this trainer can be used for the following applications:

- Spatial Orientation Training
- Situational Awareness Training
- Dynamic Flight Simulation
- Basic Flight Training
- Navigation and Weapon System Training
- Motion Sickness Desensitization
- Spatial Disorientation and Situational Awareness Research

Designed specifically for Spatial Disorientation training, the GYROLAB can simulate almost all the orientation illusions that pilots experience in flight including:

- Coriolis
- Somatogyral (Leans, Graveyard spin, Graveyard spiral)
- Oculogyral
- Somatogravic (G-Excess, Inversion)
- Oculogravic
- Vection
- Nystagmus
- False Horizon Illusions (Day and Night)
- Visual Illusions (Approach and Landing)

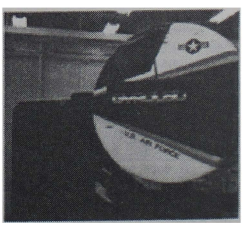


Figure 5: GYROLAB GL-2000 (Brooks AFB, TX)

Group Two consists of a new approach to motion systems that is currently under development. This system is designed to provide realistic,accurate, and economical motion cueing for spatial disorientaiton and also flight simulation. The high torque electric motor and bell crank design allows the motions system to support high payload weights, while minimizing acquisition and operation and maintenance costs. A Spatial Disorientation trainer using this new design is being constructed for the Spanish Air Force. The 4 DOF elecrtromechanical motion system is shown in figure 3.

Group Three consists of 12 trainers that are being used by the FAA and military aeromedical training centers worldwide. Cockpits are configured as a generic military training aircraft with a center stick and a single engine turboprop aeromodel. An example of this type of trainer is the GYRO Integrated Physiological Trainer (IPT) by ETC.

The GYRO IPT meets the requirement for a low cost, dynamic system that provides real-time flight simulation and can duplicate many common orientation problems experienced in flight. It can accurately and realistically simulate many orientation illusions that pilots experience in flight including:

- Coriolis
- Somatogyral Illusions (Leans, Graveyard Spin, Graveyard Spiral)
- Somatogravic illusions
- Oculogyral Illusions
- Nystagmus
- False Horizon Illusions (Day and Night)
- Runway/approach illusions
- Autokinesis

The trainer is also an excellent tool for the following training applications:

- Spatial orientation training
- Basic flight training
- Pilot Selection
- Motion sickness desensitization
- Spatial Disorientation research

Group Four consists of low fidelity single axis motion trainers used to demonstrate somatogyral vestibular illusions including coriolis and the somatogyral illusion. These trainers may have electromechanical motions systems or they may be manually powered. Two types of trainers represent this category: 1) the Barany Chair devices; and 2) the Vista Vertigon. The Barany Chair is purely a demonstrator. It does not have an aeromodel nor does it have a visual display. The trainee is a passive participant during the demonstration. An unkown number of Barany Chairs exist and are in use worldwide. The Vista Vertigon is an old technology trainer that was the first attempt to interactively involve the pilot in Spatial Disorientation training. It featured a low fidelity aeromodel and a simple visual display. Motion stimulation was limited to 360° continuous yaw stimulation only. An unknown number of Vista Vertigons are still in use today.

Group Five consists of two trainers. The design of these incorporates 360° continuous yaw motion into an existing 6 DoF hydraulic motion platform. The design places the yaw drive system, on top of the 6 DoF motion system, thereby making simultaneous motion cueing in the 6 degrees of freedom with continuous 360 degree yaw motion difficult, if not impossible. The cockpit design includes active flight controls and a visual display. The aeromodel is of a single engine turboprop aircraft. An example of this type of trainer is the DISO by AMST. Currently, one trainer has been operational for about one year and the other has just been installed. Accordingly, the training capability of this device is not yet well known although its ability to provide realistic and accurate Spatial Disorientation training is questionable.

5. HUMAN CENTRIFUGES

The value of centrifuge based acceleration training has been well established during the decades since the 1970's and has been the subject of much research and literature. Certainly, this training has allowed pilots and crews to more safely exploit the capabilities of high performance aircraft. This training will become even more important as new weapons systems are fielded with even higher performance capabilities.

Proper training for high G remains the most effective method of increasing G tolerance. This training should include two parts: Academic training on acceleration physiology and the Anti G Straining Maneuver (AGSM), and Centrifuge training.

The AGSM is a psychomotor activity which must be taught at the performance level and involves two objectives: actual performance of the AGSM and understanding when the maneuver must be initiated and performed. However, performing the AGSM in a one G environment can produce dangerously high blood pressures after only a few cycles.

The human training and research centrifuge provides the pilot the opportunity to safely practice performing the AGSM. AGSM application learning requires a high fidelity, interactive device where the air crew can first practice the AGSM and then actually fly a realistic, high G mission profile.

An AGARD report, published in 1990 provided information on centrifuges used in NATO countries[2]. The authors used data from this report and their own

knowledge of existing and in-progress centrifuges to provide data for this section.

Centrifuges can be broadly divided into single axis (planetary drive) systems and three axes (planetary, powered pitch and powered roll) systems. Single axis systems represent a simple and economical approach to training pilots how, when and how forcefully to execute the AGSM. However, flight fidelity in these systems is very low and they are best categorized as part task trainers, specifically designed to provide training to increase the pilot's G tolerance.

A variety of design approaches exist for human single axis centrifuge motion systems. These approaches can be grouped into the following categories:

- Group One Electric Motor gear box combination
- Group Two Direct Drive Electric Motor
- Group Three Hydraulic Drive System

Group 1 consists of 5 devices. An example of this type of device is the G-LAB, by ETC.

Group Two consists of one device.

Group Three consists of 0 devices.

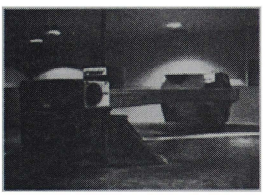


Figure 6: G-LAB Single Axis Centrifuge

Multiaxis centrifuges are more advanced centrifuges that incorporate visual displays, inrteractive flight controls and flight performance models to provide the pilot with an accurate and realistic high G flying experience. Depending on the design and motion capability devices in this category of machines may be able to support Sustained G Dynamic Flight Simulation for high performance fighter aircraft. Cockpits and aeromodels may be generic or specific, depending on the needs of the user. A state-of-the-art example of an extremely high performance, Sustained G Dynamic Flight Simulation capable centrifuge is the one being currently manufactured for the Royal Air Force in the United Kingdom.

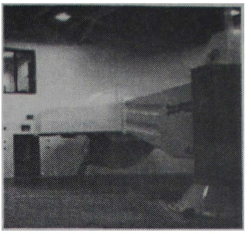


Figure 8: G-FET Dual Gimballed Centrifuge

A variety of design approaches exist for multi-axes human centrifuge motion systems exists. These approaches can be grouped into the following categories:

- Group One Electric Motor gear box combination
- Group Two Direct Drive Electric Motor
- Group Three Hydraulic Drive System

Group One consists of 4 devices. This represents the highest performance and most efficient and economical way to generate high G forces for pilot trianing. An example of this group of centrifuges is the G-FET series I and II, by ETC.

The G-FET is a multi-axis centrifuge for high G training and Sustained G Dynamic Flight Simulation. It's wide array of training capabilities include:

- G-Tolerance Improvement
- Super-Maneuverability
- Unusual Attitude Recovery
- Situation Awareness Improvement
- Missile Avoidance
- Air Combat Manoeuvres
- High-G Physiology Research

The Second Generation G-FET (currently under construction) can produce from -8Gz to + 25Gz, 0 to +8 Gy, and +/-10 Gx sustained acceleration at onsets up to 15 G/s instantaneous and 10 G/s average . The cockpit is suspended in a electromechanically powered, dual high performance gimbaled positioning system, giving the device the ability to support sustained G dynamic flight simulation. The cockpit has closed loop flight controls, and can be equipped with reconfigurable instruments and cockpit features (throttle, stick and rudder pedals), 0 to 90 degrees seat angle capability, and wide field view, out - the - window visual scenes. These systems are integrated through advanced computer and control architecture and aircraft flight algorithms to produce high fidelity, aircraft specific performance.

Group Two consists of 3 devices. A current example of this group of devices is the multi-axis centrifuge currently being installed by Wyle in Sweden for the Sweedish Air Force. Older designs in this group include the DES at Wright Paterson AFB, and a centrifuge manufactured by Austria Metal loacated in Konigsbruck, GE.

Group Three consists of 2 devices. These centrifuges were manufactured by Latecore.

6. CONCLUSIONS AND CLOSING REMARKS

There is no simple one answer to the title question. We can summarise, that motion systems in simulation devices are necessary in the following instances:

- When the motion stimuli acting on a pilot during the real flight are of primary importance;
- When we want to support the creation of the internal model of flight control in the ab initio or inexperienced pilot's mind;
- When the motion stimuli during the real flight could create disorientation or situational awareness problems to a pilot.
- When training for recovery from aircraft upset.
- When training the pilot to safely and effectively operate in a high G flight environment.

Where it is usually necessary?

- In the simulators prepared for the training centres.
- In the helicopter flight simulators.
- In the transport, bomber, tanker and Electronic Warfare aircraft flight and mission simulators.
- In the disorientation training devices.
- In the devices that must support High G training and Sustained G Dynamic Flight Simulation.

Price of motion systems remains a major consideration, but it should not be the primary one. Because of the decline of motion systems prices and the increase in their reliability and flight fidelity one should seriously consider their use in new training devices. This would also allow multiple uses of the same training device. The technology improvements in the motion base field, (e.g. electromechanical motion systems, precise computer control) will assure positive results in simulation for military pilot training.

END NOTES

[1] M. Hutny: *Military Aircraft Simulator Data Base.* ETC-PZL Aerospace Industries, Warsaw 2000. (Interim Report).

[2] AGARD AG 322, AGARD Working Group 14, High G Phsyiological Protection Training, Chap. 6, 1990.

OTHER REFERENCES

[1] AGARD-CP-588, A Review of Motion Based Physiological Training Devices, W.F. Mitchell and R. A. Leland, 1996.

[2] Gillingham Kent K and Previc Fred H. *Spatial Orientation in Flight*: Final Technical Report for the Period January 1982-January 1993, #AL-TR-1993-0022, Crew Systems Directorate, Crew Technology Division, Brooks Air Force Base, Texas, 1993.

[3] Leland, Richard A., *Spatial Orientation Training For Aircraft Operators:* AeroMedical Trianing Institute, Environmental Tectonics Corporation, 1998.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37258

AIAA 2000-4299

Advanced Helicopter Flight and Aeroelastic Simulation based on General Purpose Multibody Code

Thomas Hablowetz*
Institute of Flight Mechanics and Control; University of Stuttgart
Pfaffenwaldring 7a; 70550 Stuttgart, FRG

net: www.ifr.uni-stuttgart.de

Abstract

Modern control design and validation in helicopter technology require advanced modeling tools. The objective of this study is the provision of a simulation tool for flexible rotorcraft. The major research contribution lies in the development of an integrated structural and aerodynamic simulation environment based on proven engineering tools in computational fluid dynamics and multibody simulation. A multidisciplinary simulation environment has been established. The simulation tool allows aeroelastic analysis and controller in the loop simulation of flexible rotorcraft. Due to the underlying general purpose multibody code the structural model may easily be extended as required by the application. Exemplarily full six degree of freedom flight simulations with stability augmentation controller are presented.

Introduction

Rotorcraft applications are widely spread in aeronautics nowadays. Their unique abilities in transport, surveillance and air rescue have ensured their great success in modern society. Typical problems regarding passenger comfort and broad acceptance lie in the inherent high vibration and noise level due to the rotating lift producing mechanism. As a result all structural parts related to the rotating mechanical system have to be inspected and replaced in rather short time intervals. Due to high maintenance and operational cost but also to passenger comfort it is of great interest to further reduce vibration and interior noise. Furthermore stringent restrictions on noise level for flying in densely populated areas require further reduction of emitted sound pressure level.

Much research has been conducted to tackle these problems. Promising new methods in vibration reduction have been introduced by using active control strategies[1] like higher harmonic control (HHC),[2] individual bade control (IBC),[3] the actively controlled trailing edge flap (ACF)[4] and active control of structural response (ACSR).[5] Regarding the noise problem, control studies have been focussing on the helicopter interior noisefield,[6] whereas the area of noise emission research is still restricted to analysis and prediction of sound pressure experienced in the neighbourhood of the rotorcraft.[7] Through advances in smart structure technology the active control strategies just described become more likely to be put into practice.

It is this field of research, the smart structures and its application to aerospace, which has become of great interest at the University of Stuttgart in recent years. Various institutes are conducting research on the application of smart structures on rotorcraft. For assessing the benefit of smart structures in rotorcraft control and aerodynamics high fidelity models of the structure-fluid interaction are in demand.

In this paper the development of a modeling approach comprising structural and aerodynamic models and its application to helicopter simulation will be presented. The accuracy level of both the structural and the aerodynamic model is modularly adaptable as required by the analysis to be performed (e.g. flight simulation or aeroelastic analysis). The underlying structure and aerodynamic models are based on sophisticated engineering tools. The accuracy levels on the aerodynamic side are determined by a 'fast' blade element inflow model[8] with aerodynamic coefficients tuned by flight test result,[9] a vortex lattice method (2D discretization)[10] and a 3D Euler method.[12] On the structural side there is a choice between a purely rigid body model, elastic beam models for the rotor blades and arbitrary FEM

*Research Assistant.
email: thomas.hablowetz@po.uni-stuttgart.de
Copyright ©2000 by T. Hablowetz. Published by the American Institute of Aeronautics and Astronautics Inc, Inc., with permission.

models for both blades and fuselage. The overall simulation model is based on the general purpose multibody code SIMPACK,[14] developed at DLR (German Aerospace Research Center). As an option, controller simulation engines such as Simulink or SystemBuild can be linked. In case of structural modeling based on FEM (NASTRAN), the FE model is included in a preprocessing step. Complex aerodynamic models are provided via Inter Process Communication (IPC) standard interfaces to CFD codes, which have been developed at the Institute of Aerodynamics at Stuttgart University.[10,12,13]

Examples of multibody modeling in helicopter simulation can be found in[21-23] with particular focus on rotor dynamics. An application of MBS simulation to identification of controller design models is reported in.[24] All these works are mainly based on analytical inflow models as reviewed in,[25] which of course is reasonable and sufficient for real-time simulation purposes. Usage of MBS in aeroelastic analysis with basic aerodynamic wake modelling is introduced by.[26-28]

In the following the concept of a modular simulation tool for helicopter application based on general purpose multibody simulation code is presented. Having reviewed the basic ideas of multibody simulation the modular concept of model refinement on both the structural and the aerodynamic part will be discussed. For the aerodynamics and the controller part the modular concept means the coupling of specialized engineering tools on the level of data exchange via inter process communication.

Furthermore the MBS model of a 5 blade Hughes 500 helicopter will be presented. The tuning of "fast aerodynamics" based on analytical inflow models is conducted by CFD calculations. Finally some 6 dof flight simulation will be presented including a basic stability augmentation (SAS) controller.

Simulation tool based on General Purpose Multibody Simulation Code

Multibody Simulation

Multibody systems (MBS) are models of technical systems consisting of rigid or deformable bodies. The bodies contain mass, inertia and geometrical properties. They are connected to each other or to the environment by means of joints and force interaction. The environment may be an arbitrary moving reference frame or just inertial fixed. Joints denote the restriction of each body to move with atmost 6 degrees of freedom to each other depending on restrictions defined by neighbouring bodies. Force interaction denotes the force interference of such bodies or the environment, e.g. gravity or aerodynamic drag. A general multibody system, as considered here, is shown in figure 1. A method to provide differential equations to describe the MBS behaviour is called a multibody formalism.

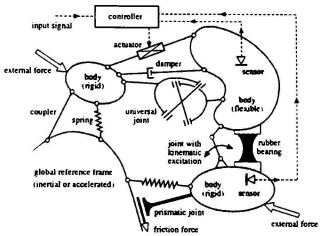


Figure 1: General Multibody System

Multibody formalisms are computer algorithms to generate automatically the equations of motion for systems of the general form shown in figure 1. These are based on data, which describe the system elements and system topology, i.e. the way the nodes on the system bodies are connected by force elements and joints. Two groups of formalisms may be distinguished resulting in basically different types of equations of motion. The first group yields the Lagrangian equations of type 1, which contain the unknown generalized constraint forces in terms of Lagrangian multipliers. These differential equations are accompanied by a set of algebraic constraint equations. The resulting representation of the system motion is sometimes called the descriptor form of the equations of motion. It is simple to generate, but it requires the numerical solution of differential-algebraic equations.[15] By contrast, the second group of formalisms provides the state space representation of motion, i.e. a minimal set of first order (kinematical and dynamical) differential equations, in which the constraint forces have been eliminated. Numerical methods for solving these equations are often considered to be more mature with respect to computational efficiency. The starting point for the development of both types of formalisms are the equations describing the motion of a representative system body i, acted upon by the applied external and internal forces and torques due to the force elements and the unknown internal joint forces and torques between the system bodies.

Kinematics The motion of an arbitrary body is described by its position ($\mathbf{x_I}$) and its velocity ($\mathbf{x_{II}}$) vector:

$$\mathbf{x_I} = \begin{bmatrix} \mathbf{r} \\ \alpha \\ \mathbf{q} \end{bmatrix} \qquad \mathbf{x_{II}} = \begin{bmatrix} \mathbf{v} \\ \omega \\ \dot{\mathbf{q}} \end{bmatrix} \tag{1}$$

These vectors satisfy the kinematic equations of motion

$$\dot{\mathbf{x}}_\mathbf{I} = \mathbf{X}(\mathbf{x_I}) \cdot \mathbf{x_{II}}\,, \quad \mathbf{X}(\mathbf{x_I}) = \begin{bmatrix} \mathbf{E} & \tilde{\mathbf{r}} & \mathbf{0} \\ \mathbf{0} & \mathbf{X}_{ang} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{E} \end{bmatrix} \tag{2}$$

which exhibit the linear dependency of the derivatives of the position variables with respect to time on the velocity variables.

Kinetics The general intend of formulating equations of motion in computational dynamics is not to find nicely problem-adopted equations, moreover it is aimed to formulate the algorithm in a way to scope with a broad variety of model classes in regard to computational efficiency.

Based on Hamilton's principle one yields:

$$\mathbf{M}\dot{\mathbf{x}}_{II} = \mathbf{h_a} + \mathbf{h_c} \tag{3}$$

The matrices $\mathbf{M}$, $\mathbf{h_a}$ and $\mathbf{h_c}$ denote the generalized mass matrix, the applied and constraint forces respectively. The applied force may be separated into:

$$\mathbf{h_a} = \mathbf{h}_\omega + \mathbf{h_g} + \mathbf{h_e} + \mathbf{h_p} + \mathbf{h_f} \tag{4}$$

In this expression $\mathbf{h}_\omega$ are generalized inertia forces due to angular velocity ω of the body reference frame motion. They are as well as the gravitational forces $\mathbf{h_g}$ distributed over the body volume V_0. The generalized internal forces $\mathbf{h_e}$ result from elastic body deformation whereas $\mathbf{h_p}$ is due to external surface forces. The last term $\mathbf{h_f}$ represents the forces and moments applied by force elements attached to the body, they are known functions of the system states and possibly additional quantities as shown in fig. 1. Forces arising from joints are unknown, they yield the constraint forces. The generalized mass matrix M can be partitioned according to the $6 + n_e$ velocity variables $\mathbf{x_{II}}$:

$$\mathbf{M}(\mathbf{x_I}) = \begin{bmatrix} m\mathbf{E} & \dots & \text{sym.} \\ m\tilde{\mathbf{c}} & \mathbf{I} & \dots \\ \mathbf{M}_{et} & \mathbf{M}_{er} & \mathbf{M}_{ee} \end{bmatrix} \tag{5}$$

The scalar m represents the body mass, matrices c and I stand for the distance vector of centre of mass from body reference frame and the inertia tensor of the deformed body respectively. The sub-matrix $\mathbf{M_{ee}}$ contains the generalized masses with respect to the modal coordinates q, arising from Ritz approximation accounting for elastic bodies. The matrices $\mathbf{M_{et}}$ and $\mathbf{M_{er}}$ contain the coupling terms of reference motion and deformation, respectively.

The separation of body motion into reference motion and deformation leads to a corresponding separation of linear and angular momentum vector for the body. All of the generalized forces and masses in equation (3) are algebraic expressions, containing the state variables (1) and integrals over the shape functions.[16]

Having derived kinematic and kinetic equations for one body benefit is drawn of the general tree structure of mechanical systems. Thus the equations can be applied to each body coupled by constraint equation restricting relative motion among them. Systems with kinematic loops are transformed to tree structured systems. The loop closing constraints, obtained as algebraic equations, form the Differential Algebraic Equations (DAE). Special adopted solvers are developed to scope these problems.[15]

Application examples as considered here appear to be of tree structure. The implemented recursive equation set up scheme yields the nonlinear equations of motion in explicit form

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t) \tag{6}$$

where $\dot{\mathbf{x}}$ are the generalized states (position and velocity) of the system. The matrix $\mathbf{u}$ denotes inputs to the system. Note the benefit in computational efficiency of the so called $O(n)$ formalism to generate explicit ODE by avoiding inversion of the overall system mass matrix[17] (processing time increases linear with the number of bodies).

MBS Interfaces via IPC Coupling

Adressing multi-field problems such as the structure-fluid coupling of elastic aircraft, the underlying MBS code SIMPACK offers the possibility to interfere when creating the equation of motion (6). This can be achieved by means of so called User Routines allowing for coding of user defined functionality. Regarding equation (4), this corresponds to introduce user defined forces $\mathbf{h_f}(\mathbf{x}, \mathbf{u}, \mathbf{t})$ to the system. At the same time the full state vector $\mathbf{x}$ is made available. These are exactly the values needed to match the boundary conditions of each field considered: The mechanical system, defined by a set of rigid and flexible bodies is been loaded by nodal forces and torques to approximate the continuous distributed fluid forces. The fluid field with its common contact

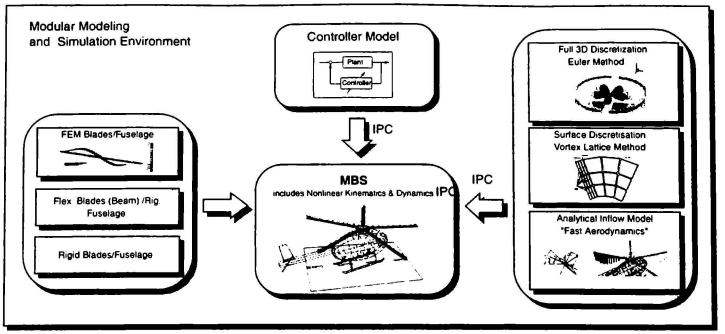


Figure 2: Modular Modeling and IPC Simulation Interfaces

areas to the surface of the bodies has to fulfill the kinematic boundary conditions given by the body surface position and velocities. This requires the choice of fluid-structure contact surfaces (wet surface) for which a nodal discretization has to be done. For these nodes, kinematics are made available to the fluid solver, which calculates the resultant nodal force and torque load.

Considering a typically coupled problem such as an aircraft wing in free flow conditions, the discretization and solving of the fluid grid requires a far larger amount of processor power and memory consumption than the participated structural solver. For this reason, the Panel- and Euler - fluid solvers have to run on high performance multiprocessor computers to guarantee results in reasonable time. The MBS code as the structural solver requires less than one percent of overall simulation time and runs easily on standard Unix or PC workstations. To enable necessary software communication, an Inter Process Communication (IPC) scheme had been developed and set up.[13] It enables platform independent data communication via Internet. Another important interface is the possibility of having linked the MBS code to Control system analysis programs. A far simpler case of a coupled multi-field problem is that of controller-MBS interference, some output or state quantities of the mechanical system are measured and feed back by a mostly linear feedback law to generate actuator signals. An important issue is the possible introduction of algebraic loops to the overall simulation by direct-feed-through terms. This has to be accounted for choosing a numerical solver algorithm. For generality and simplicity of the overall simulation scheme, the same interface method via IPC had been used to link control loops established in MatrixX.

Modular structural modeling

Depending on the application example considered different level of complexity are possible (see fig.). The simplest one is the purly rigid case. All bodies in the helicopter model are selected to be rigid. This might be sufficient for trim calculations and necessary when using the overall simulation for real time simulation purpose. The next stage of complexity is given by selection of flexible blades. Hereby either simple to model Euler-Bernoulli beams are available or arbitrary complex beam or shell models for refined FE-modelling of the blades. The flexible bodies are hereby set in a preprocessing step, in case of FE modelling a modal analysis has to be performed. The highest level of model complexity is given by full FE modeling of both the rotor blades and the fuselage. This might be necessary, investigating vibration level inside the cabin at the pilot' seat or at location of sensitive payload.

Methods of modeling flexible bodies in a multibody system have been reviewed in.[18] Here the floating frame of reference formulation will be used. In this methodology the motion of a flexible body is subdivided into a reference motion and a deformation. The former may be described as the motion of a body reference frame, whereas deformation is the motion of the points of the body with respect to its reference frame. Introducing a Ritz approximation, one obtains a representation of the body deformation by a reduced set of modal variables.[19]

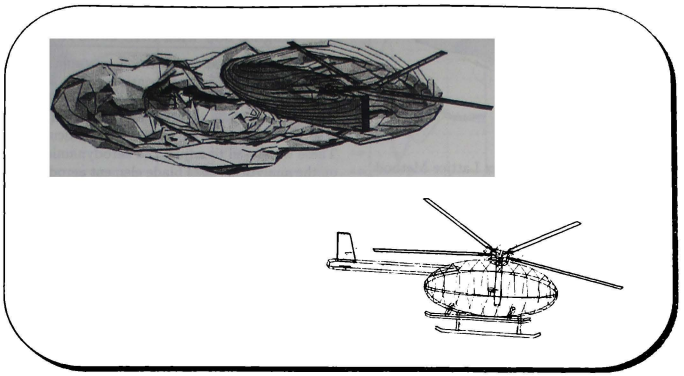


Figure 3: Coupled simulation showing free wake (partially) of a rotor in forward flight after three revolutions (Vortex Lattice Method coupled with flexible MBS)

The deformations are assumed to be small which holds for many applications. Simplifications due to linearization can be applied to increase computational efficiency. In case of high acceleration, e.g. due to high rotational velocity (helicopter application), high inertial forces act upon the body. If the stiffness in direction of inertia load is high, the system deformation remains small. However, in this case additional terms in the linearised equations have to be considered, so called 'geometric stiffness terms'[16,20] which are accounted for in the MBS code.

Modular aerodynamic modeling

For the coupled aeroelastic and flight mechanical simulation different stages of accuracy (and also processing speed) of aerodynamic models can be chosen (fig. 2).

For trim and 6 dof flight simulation an analytical blade element theory is available. The lift coefficients are either given from tables or might be adjusted (tuned) by simulation with aerodynamic models of higher level of accuracy. The data for drag are estimated from profile data. The basic task using an analytical approach lies in the determination of the local induced velocity. The method applied here is the calculation of a mean induced velocity for the rotor disk fulfilling mass, energy and momentum conservation for the rotor as an entity (momentum theory). This results in a radial constant mean induced velocity. The combination of momentum and blade element theory gives a radial distribution of induced velocity and allows for consideration of local geometrical and aerodynamical parameters. By means of user defined force elements, this method has been directly implemented in the MBS code.[8] Further work is in progress to include dynamic inflow models[25] to improve this fast method.

The next level of accuracy is defined using a panel method, the 'Rotor Free Wake Vortex Lattice Method' (ROVLM).[10] This panel code follows linear velocity potential theory. The doublet strength of each new spanwise wake row at the end of each time step is obtained from the blade trailing edge panels of each spanwise section. The ROVLM code had to be modified to use the common IPC interface data for kinematics of the 'wet surface' nodes as well for the rigid body motion of rotor and fuselage of the MBS. Thus the actual rotor geometry and its full velocity state is generated from the kinematic data. Having resolved pressure from velocity distribution, local force vectors for each blade panel can be calculated. These are transfered back to the MBS coupling nodes to apply aerodynamic load.

Basics of the method and its modification can be found in.[10,11] The panel method has been used to tune aerodynamic parameters (figure 3). possible application of simulation with controllers using active trailing edge flap has been prepared (figure 4).

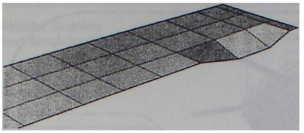


Figure 4: Flap model in Vortex Lattice Method

The highest level of accuracy made available consists in coupling the MBS model to a finite-volume Euler method called INROT.[12,13] The physical laws of conservation of mass, momentum and energy constitute the founding equation for all aerodynamic equation. Applying these equations to an infinite small control volumina, one yields a system of nonlinear partial differentiel equations, which are well known as the Navier Stokes Equations. Its solution for practical problems is quite difficult in terms of processing time and memory requirement. Neglecting effects such as friction and heat transfer the equation simplify to the Euler equations. Regard to parallel processing, INROT uses the so called Chimera technique to discretisize the 3D fluid field. This technique allows computational efficient discretization of the field in case of relative motion among different aerodynamic bodies. In case of the helicopter application there are individual grids around each blade and the fuselage (fig. 5).



Figure 5: Chimera grid of Hughes 500 rotor

All individual body grids move in a base grid which covers the entire computational domain. In contact regions of the grids the boundary conditions are fulfilled. Reference simulations have been conducted,[13] Aeroelastic investigations on rotary and fixed wing applications are in progress.

Application Example: 6dof Simulation of Hughes 500 with SAS controller

In the following an application example will be presented. The helicopter considered is a 5 bladed Hughes 500 helicopter. From literature[9] necessary data such as blade bending and loads were available to conduct coupled MBS-CFD calculations. These were used to adjust the aerodynamic values on the analytical (fast) blade element aerodynamic in order to perform 6 dof flight simulation.

The multibody modell consist of 19 rigid and 5 elastic bodies (rotorblades). Some of the bodies form rigid entities such as the fuselage and the tail boom body. The animation modell is shown in figure 3. The fuselage body is connected to the inertial system, the choice of the fuselage joint determins the degrees of freedom in respect to the fligth mechanic motion. The blades are fully articulated with flap, lag and pitch joints. There is no swashplate modelled since there were no intention to examine control loads. The blade pitch control is driven directly by torque force elements allowing direct or, if desired, individual blade control. The model comprises 21 rigid body degree of freedom, additional 3 elastic modes (2 bending, 1 torsion) from each blade, in total 72 first order ODE states. Mass and geometrical properties of the rigid bodies had to be estimated, since only data about overall mass and geometry were available. Special care had to be done modeling the flexible blades, total mass and lowest eigenfrequencies are identical to given data.[9]

Like any aircraft in steady flight, the helicopter must be in equilibrium with respect to three forces and moments. To maintain a desired flight condition it is necessary to determine adequate control inputs. In a first step, simplified trim equations are solved for hover and forward flight to give estimates of trim-states and controls. Applying these values to the 6dof simulation with

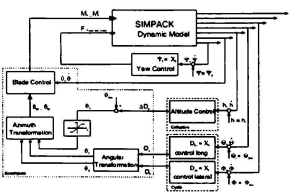


Figure 6: Basic SAS controller structure

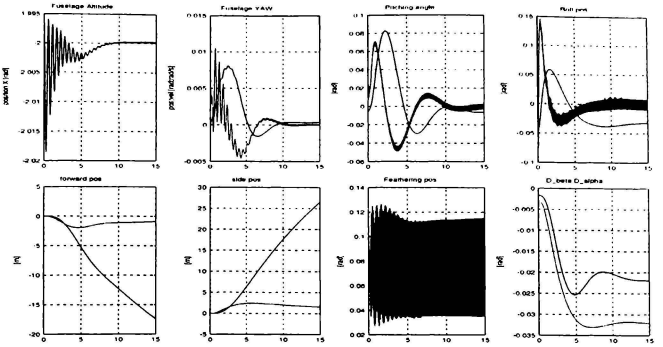


Figure 7: States history from near trim state (SAS controller on)

stability augmentation control loops implemented yield the proper trim values, since the deviation in the angular rates are compensated. The different control loops set up in MatrixX are shown in figure 6.

Besides the basic SAS controller (attitude control) given by rate feedback (roll, pitch), control loops for yaw and altitude hold are implemented. The blocks denote PD and PID controller structure respectively. The gains are determined by simplified second order differential equations for rigid body motion (assumption of decoupling of axis) and are adjusted empirically by simulation. Figure 7 gives an example for state values of the fuselage along with the control inputs D_i. It can be seen, that the SAS controller stabilizes the rotorcraft except the lateral position, since no lateral motion control is implemented. The constant state and control values are required trim values for the simulation.

Summary and Conclusions

At the University of Stuttgart research on smart structures and their application to aircraft has been conducted since several years. Specific needs for controller design and verification but also for basic studies of physical phenomena in rotorcraft application has driven further development of simulation and modeling capabilities. This paper presents a simulation environment for flexible rotorcraft arising from these demands. The tool is a modular ensemble of proven software tools, each of them highly specialized in their own engineering discipline. The center link to all modules is a general purpose multibody code. Depending on the issue to be investigated, the structural model can be refined and extended by finite element models of certain bodies. Regard to aerodynamic phenomena to be considered, the aerodynamic model can be varied in certain stages.

Having reviewed the basic ideas of multibody simulation the modular concept of model refinement on both the structural and the aerodynamic part were presented. In the sense of aerodynamics and controller link the modular concept is realized on the level of data exchange via inter process communication. Remarkable advantages are decentralized calculation, maintainance and development of the participated software tools. Finally an application example was presented, the MBS model of a 5 blade Hughes 500 helicopter with stability augmentation controller.

At present calculations of rotor-vibration controller are in preparation The aim of the controller is to diminish rotor hub vibrations, therefore at least the rotor blades have to be considered as flexible. Since the wake field of the rotor is the main source of unwished vibrations, a high accurate aerodynamic model is required. Another application is the validation of the simulation tool towards flutter phenomena. Besides rotary wing also fixed wing configuration are under examination.

References

[1] Friedmann, P.P., Millott, T.A. *"Vibration Reduction in Rotorcraft Using Active Control: A Comparison of Various Approaches"*, Journal of Guidance, Control and Dynamics, Vol. 18, No. 4, 1995, pp. 664-673.

[2] Richter, P., Schreiber, T. *"Theoretical Investigations and windtunnel tests with HHC-IBC"*, Proceedings 20th European Rotorcraft Forum, Amsterdam, Paper No. 71, 1994.

[3] Nitzsche, F. *"Designing efficient helicopter individual blade controllers using smart structures"*, Proceedings AIAA/ASME Adaptive Structures Forum, Hilton Head, SC., 1994.

[4] Myrtle, T.F., Friedmann, P.P., *"Vibration Reduction in Rotorcraft Using the Actively Controlled Trailing Edge Flap and Issues Related to Practical Implementation"*. Presented at the AHS 54rd Annual Forum, Washington, D.C., 1998.

[5] Chiu, T., Friedmann, P.P., *"ACSR system for vibration suppression in coupled helicopter rotor/flexible fuselage model"*, AIAA/ASME/ASCE/AHS/ASC Structures. Structural Dynamics and Materials Conference and Exhibit, 1996, Technical Papers. Pt. 4 (A96-26801 06-39), 1996, pp. 1972-1990.

[6] Gembler, W., Schweitzer, H., *"Helicopter Interior Noise Reduction by Active Gearbox Struts"*, 54th AHS Forum, Washington, D.C., May 1998.

[7] Zerle, L., Wagner, S., *"Direct BVI-Sound Pressure Calculation by Using Aerodynamic Data and Application of Retarded Potentials."*, In Proceedings of the 25. European Rotorcraft Forum, Rome, Italy, September 1999, Paper No. B9.

[8] Martin , G., *"Aufbau und Validierung eines Rotor-Mehrkörpermodells."*, Diploma Thesis, IFR_SR_97.004, Institut of Flightmechanics and Control , University of Stuttgart, 1997.

[9] Lindert, H.-W., *"Anwendung einer strukturmechanischen Methode zur Rekonstruktion der Luftkräfte am rotierenden Rotorblatt aus Windkanal und Flugversuchsmeßdaten."*, Ph.D. Thesis, RWTH Aachen, VDI-Fortschrittsberichte, Reihe 12, Nr. 245, VDI Verlag, Düsseldorf, 1994.

[10] Zerle, L., Wagner, S., *"Development and Validation of a Vortex Lattice Method to calculate the Flowfield of a Helicopter Rotor Including Free Wake Development."*, In Proceedings of the 18. European Rotorcraft Forum, Avignon, France, September 1992, Paper No. 72.

[11] Lange , H., *"Kopplung eines potentialtheoretischen Panelverfahrens mit einem Mehrkörper simulationsprogramm zur Berechnung der Interaktion wischen Strömung und Struktur bei einem Hubschrauberrotor."*, Diploma Thesis, IFR_SR_99.014, Institut of Flightmechanics and Control , University of Stuttgart, 1997.

[12] Stangl. R., *"Ein Eulerverfahren zur Berechnung der Stömung um einen Hubschrauber im Vorwärtsflug."*, Ph.D. Thesis, Institut fr Aero- und Gasdynamik, Universität Stuttgart, 1996.

[13] Hierholz, K.-H., Wagner, S., *"Simulation of Fluid-Structure Interaction at the Helicopter Rotor."*, ICAS-98-2.5.1, Melbourne, Australia, September 1998, Paper No. B9.

[14] Kortüm, W., Rulka, W., Eichberger, A., *"Recent Enhancements of SIMPACK and Vehicle Applications."*, European Mechanics Colloquium, EUROMECH 320, Prague, 1994.

[15] Eich-Soellner, E. and Führer C., *" Numerical Methods in Multibody Dynamics "*, Teubner, Stuttgart, 1998.

[16] Schwertassek, R., Wallrapp, *Dynamik flexibler Mehrkörpersysteme.*, Vieweg & Sohn Verlagsgesellschaft mbH, Braunschweig/Wiesbaden, 1999.

[17] Schwertassek, R., Rulka, W. *Aspects of Efficient and Reliable Multibody System Simulation.*, Real-time Time Integration Methods for Mechanical System Simulation, pp. 55-96, E.J. Haug, R.C.Deyo (eds.), Springer Verlag, Berlin, 1991.

[18] Shabana, A.A., *" Flexible multibody dynamics: review of past and recent developments "*, Multibody System Dynamics 1, 1997,pp. 189-222.

[19] Meirovitch, L. and Kwak, M. K., *" On the modeling of flexible multi-body systems by the Rayleigh-Ritz method"*, , AIAA Dynamics Specialists Conference, Long Beach, CA, April 1990.

[20] Schwertassek, R.: *Flexible bodies in multibody systems.*, Computational Methods in Mechanical Systems, pp. 329-363, J. Angeles and E. Zakhariev (eds.), Springer-Verlag, Berlin, 1997.

[21] Elliott, A.S., McConville, J.B., *"Application of a General Purpose Mechanical Systems Analysis Code to Rotorcraft Analysis problems."*, AHS Specialists' Meeting on Rotorcraft Dynamics, 1989.

[22] Bachau, O.A., Kang, N.K., *"A multibody Formulation for Helicopter Structural Dynamic Analysis."*, Journal of the American Helicopter Society, 1993.

[23] Bachau, O.A., Lee, M., *"Multi-Body Formulation for Rotorcraft Dynamic Analysis."*, 51th Annual Forum of the American Helicopter Society, Fort Worth, Texas, 1995.

[24] Bertogalli, V., Bittanti, S.; Lovera, M., *"Simulation and identification of helicopter rotor dynamics using a general-purpose multibody code."*, Journal of the Franklin Institute, v 336 n 5, pp. 783-797, 1999.

[25] Gaonkar, G.H., Peters, D.A., *"A review of dynamic inflow modeling for rotrocraft flight dynamics."*, Vertica, 12(3), pp. 213-242, 1988.

[26] Lanz, M., Mantegazza, P., Ragani, P.V., *"Aeroelastic rotor dynamics by general finite element and multibody approaches"*, IX World Congress of the Theory of Machines and Mechanisms, vol. 2, pp. 1650-1656, 1995.

[27] Ghiringhelli, G.L., Masarati, P., Mantegazza, P., *"Multi-body aeroelastic analysis of smart rotor blades, actuated by means of piezo-electric devices"*, CEAS Int. Forum on Aeroelasticiy and Structural Dynamics, vol. II, pp. 115-122, 1997.

[28] Ghiringhelli, G.L., Masarati, P., Mantegazza, P., Nixon, M.W. *"Multibody Analysis of a Tiltrotor Configuration"*, Nonlinear Dynamics, vol. 19, pp. 333-357, 1999.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37259

AIAA-2000-4302

A Review of Attitude Kinematics for Aircraft Flight Simulation*

W. F. Phillips†
C. E. Hailey‡
Utah State University, Logan, Utah 84322-4130
G. A. Gebert§
Sverdrup Technology, Inc., Eglin AFB, Florida 32542

Abstract

This paper presents the development of the aircraft kinematic transformation equations in terms of four different attitude representations, including the well-known Euler angles, the Euler-axis rotation parameters, the direction cosines and the Euler-Rodrigues quaternion. The emphasis of the paper is directed at the application of the quaternion formulation to aircraft flight simulation. Results are presented which reinforce the observation that the quaternion formulation, typically implemented to eliminate the singularities associated with the Euler angle formulation, is far superior to the other commonly used formulations based on computational efficiency alone. A development of the quaternion constraints necessary to independently constrain roll, pitch, yaw, bank angle, elevation angle, and/or azimuth angle is presented. For verification of simulation codes, a general closed-form solution to the quaternion formulation, for the special case of constant rotation, is also presented. Additionally, the paper provides a discussion of the numerical integration of the quaternion formulation. This discussion is especially important for simulations that may still utilize a common error reduction scheme originally developed for analog computers. The paper includes a detailed review of the literature on attitude representation dating from the early work of Euler and Hamilton to recent publications in fields such as navigation and control.

Nomenclature

$\mathbf{A}$	=	arbitrary quaternion
$\mathbf{B}$	=	arbitrary quaternion
$[\mathbf{C}]$	=	direction-cosine matrix
D	=	temporal differential operator
$\bar{\mathbf{E}}$	=	Euler axis vector
E_x	=	x-component of Euler axis vector
E_y	=	y-component of Euler axis vector
E_z	=	z-component of Euler axis vector
$\mathbf{e}$	=	Euler-Rodrigues quaternion
$\mathbf{e}^*$	=	conjugate of Euler-Rodrigues quaternion
e_0	=	scalar component of Euler-Rodrigues quaternion
e_x	=	x-component of Euler-Rodrigues quaternion
e_y	=	y-component of Euler-Rodrigues quaternion
e_z	=	z-component of Euler-Rodrigues quaternion
g	=	acceleration of gravity
$\bar{\mathbf{i}}_x$	=	unit vector in the x-direction, earth-fixed coordinates
$\bar{\mathbf{i}}_y$	=	unit vector in the y-direction, earth-fixed coordinates
$\bar{\mathbf{i}}_z$	=	unit vector in the z-direction, earth-fixed coordinates
$\bar{\mathbf{i}}_{x_b}$	=	unit vector in the x_b-direction, body-fixed coordinates
$\bar{\mathbf{i}}_{y_b}$	=	unit vector in the y_b-direction, body-fixed coordinates
$\bar{\mathbf{i}}_{z_b}$	=	unit vector in the z_b-direction, body-fixed coordinates
k	=	gain coefficient
p	=	rolling rate, body-fixed coordinates
q	=	pitching rate, body-fixed coordinates
r	=	yawing rate, body-fixed coordinates
$\mathbf{Q}$	=	arbitrary quaternion
$\mathbf{Q}^*$	=	conjugate of arbitrary quaternion
Q_0	=	scalar component of arbitrary quaternion
Q_x	=	x-component of arbitrary quaternion
Q_y	=	y-component of arbitrary quaternion
Q_z	=	z-component of arbitrary quaternion
$\mathbf{T}$	=	temporary quaternion
T_0	=	scalar component of temporary quaternion
T_x	=	x-component of temporary quaternion
T_y	=	y-component of temporary quaternion
T_z	=	z-component of temporary quaternion
t	=	time
$\bar{\mathbf{V}}_w$	=	wind vector

*Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.
† Professor, Mechanical and Aerospace Engineering Department. Member AIAA.
‡ Associate Professor, Mechanical and Aerospace Engineering Department. Associate Fellow AIAA.
§ Technical Fellow, Aerodynamics. Senior Member AIAA.

V_{wx} = x-component of wind , earth-fixed coordinates

V_{wy} = y-component of wind , earth-fixed coordinates

V_{wz} = z-component of wind , earth-fixed coordinates

$\bar{V}$ = airspeed vector

u = axial or x_b-component of airspeed, body-fixed coordinates

v = sideslip or y_b-component of airspeed, body-fixed coordinates

w = normal or z_b-component of airspeed, body-fixed coordinates

$\bar{v}$ = arbitrary vector

v_{x_b} = x-component of arbitrary vector, body-fixed coordinates

v_{y_b} = y-component of arbitrary vector, body-fixed coordinates

v_{z_b} = z-component of arbitrary vector, body-fixed coordinates

v_x = x-component of arbitrary vector, inertial coordinates

v_y = y-component of arbitrary vector, inertial coordinates

v_z = z-component of arbitrary vector, inertial coordinates

x = x-coordinate, inertial coordinates

y = y-coordinate, inertial coordinates

z = z-coordinate, inertial coordinates

x_b = x-coordinate, body-fixed coordinates

y_b = y-coordinate, body-fixed coordinates

z_b = z-coordinate, body-fixed coordinates

δt = time step

ε = orthogonality error

$\bar{\omega}$ = angular velocity vector

Θ = total rotation angle

φ = general Euler angle

ϕ = Euler bank angle or roll attitude

θ = Euler elevation angle or pitch attitude

ψ = Euler azimuth angle or heading

Introduction

The numerical simulation of motion with six degrees-of-freedom (6-DOF) is important in many areas of modern technology, ranging from computer animated film making to the development of aircraft and spacecraft flight simulators. For these varied applications, there is a need to describe both position and orientation in either an inertial coordinate system or a non-inertial coordinate system. With specific application to aircraft flight, one common method for describing attitude is through the use of Euler angles. The Euler angle formulation contains a singularity known as *gimbal lock*. As a result, alternative methods for relating non-inertial coordinates to inertial coordinates have been developed. One such formulation employs quaternion algebra to remove the singularity.

Several mechanics textbooks introduce quaternions as part of the treatment of generalized rigid-body motion.[1,2] Quaternions are applied to a diverse set of rigid-body motion problems ranging from simulations of nanotechnology,[3] underwater tethers,[4] vehicle navigation,[5,6] robotics,[7-11] molecular dynamics,[12] electromagnetic fields[13] to computer visualization and animation.[14-16] In the early 1990's, Chou[17] published a review article on quaternion kinematics for the robotics community. Also published in the early 1990's, Shuster[18] provides a review of vector/matrix algebra and attitude representations used by the spacecraft community including Euler angles, direction cosines, axis-azimuth representations, Euler-Rodrigues symmetric parameters, Rodrigues or Gibbs parameters, and Cayley-Klein parameters.

What is lacking is a current, thorough review of attitude kinematics directed primarily toward the aircraft simulation and modeling community. To support this point of view, a survey of recent and classic introductory atmospheric flight mechanic textbooks shows no[19-31] or very limited[32,33] discussion of quaternions, despite the wide use of the quaternion formulation in simulating aircraft motion.[34,35] The need for a review of attitude kinematics for the aircraft community is underscored by the use of quaternion integration schemes employed on today's digital computers, which were developed originally for analog simulations in the 1950's and early 1960's. An early application of quaternions to analog computer simulation of aircraft and missile motion was developed by Robinson[36,37] and popularized by Mitchell and Rogers.[38] Since these analog computer simulations were inherently first-order, the raw quaternion formulation did not work well. When such simulations were run, the magnitude of the quaternion would grow quite rapidly with time. Since the formulation requires the magnitude of the quaternion to remain unity, these large errors would render the simulations almost useless. To remedy this problem, techniques were developed to reduce this computer error. One error reduction scheme, due to Corbett and Wright,[39] worked quite well when used with the analog computers of that era. It is no longer necessary with the higher-order integration techniques available on today's digital computers. However, the Corbett-Wright correction is still presented as necessary in the development of quaternion kinematics in recently published aeronautics textbooks.[33,40]

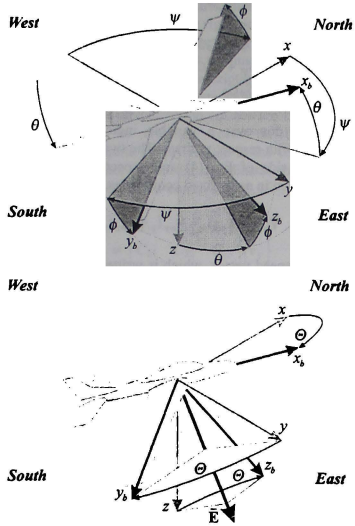


Fig. 1. Comparison between the Euler angle rotations and the Euler axis rotation.

In simulating 6-DOF aircraft motion, Newton's second law is most conveniently written in terms of a non-inertial coordinate system fixed to the moving aircraft. Position and orientation are most conveniently described in terms of an *earth-fixed* coordinate system, which for all practical purposes can be considered to be an inertial coordinate system. Here the inertial or earth-fixed coordinate system will be designated (x, y, z) and the non-inertial or *body-fixed* coordinate system will be designated (x_b, y_b, z_b). The orientations of these two coordinate systems relative to the earth and the aircraft are shown in Fig. 1. The position of the aircraft is specified by the location of the origin of the non-inertial frame with respect to the inertial frame and the orientation of the aircraft is specified in terms of the orientation of the non-inertial frame relative to the inertial frame. To develop a complete 6-DOF formulation, some means of transforming arbitrary vectors between non-inertial coordinates and inertial coordinates is required.

The Euler Angle Formulation

The orientation of the non-inertial reference frame relative to the inertial reference frame can be described in terms of three consecutive rotations through three *body-referenced Euler angles*. There are twelve possible ways to define three independent body-referenced Euler angles. Starting from the inertial reference frame, three consecutive rotations are performed, each about one of the three body-referenced axes, to arrive at the final non-inertial reference frame. The only restriction required to provide three degrees of freedom is that no two consecutive rotations can be about the same axis. With this restriction, there are six symmetric sets of Euler angles,

$$\varphi_x \to \varphi_y \to \varphi_x, \qquad \varphi_x \to \varphi_z \to \varphi_x,$$
$$\varphi_y \to \varphi_x \to \varphi_y, \qquad \varphi_y \to \varphi_z \to \varphi_y,$$
$$\varphi_z \to \varphi_x \to \varphi_z, \qquad \varphi_z \to \varphi_y \to \varphi_z$$

and six asymmetric sets,

$$\varphi_x \to \varphi_y \to \varphi_z, \qquad \varphi_x \to \varphi_z \to \varphi_y,$$
$$\varphi_y \to \varphi_x \to \varphi_z, \qquad \varphi_y \to \varphi_z \to \varphi_x,$$
$$\varphi_z \to \varphi_x \to \varphi_y, \qquad \varphi_z \to \varphi_y \to \varphi_x$$

Any of these twelve Euler angle sets can be used for attitude representation. For example, orbital mechanics and the quantum theory of angular motion use primarily the $\varphi_z \to \varphi_x \to \varphi_z$ formulation.[41,42] The rotation matrix for each of the twelve sets of Euler angles are presented by Tandon.[43] Graphical methods to illustrate the Euler angle transformations are also available.[44-46]

The aeronautics community commonly uses the last of the above rotation sequences, $\varphi_z \to \varphi_y \to \varphi_x$. These three Euler angles, commonly written ψ, θ, and ϕ, are respectively the azimuth angle or heading, the elevation angle or pitch attitude, and the bank angle or roll attitude. For this particular set of Euler angles, the orientation of the body-fixed frame, (x_b, y_b, z_b), relative to the earth-fixed frame, (x, y, z), is described by performing the three consecutive rotations in the specific order as follows. First, the earth-fixed frame is rotated about the z-axis through an angle, ψ. Next, the revolved reference frame is rotated about the new y-axis through an angle, θ. Finally, the revolved reference frame is rotated about the new x-axis through an angle, ϕ, to arrive at the final body-fixed reference frame. This set of Euler angles is shown in Fig. 1. Readers should be cautioned that other rotation sequences have

been associated with the aeronautics community, for example $\varphi_z \rightarrow \varphi_x \rightarrow \varphi_y$.[47]

With the usual Euler angle definitions, consider an arbitrary vector $\bar{v}$ having components (v_x, v_y, v_z) in the inertial coordinate system and having components $(v_{x_b}, v_{y_b}, v_{z_b})$ in the non-inertial coordinate system. Using the notation $S_\chi = \sin(\chi)$ and $C_\chi = \cos(\chi)$, the transformation of this vector from inertial coordinates to non-inertial coordinates is given by[29,43,48,49]

$$\begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\phi & S_\phi \\ 0 & -S_\phi & C_\phi \end{bmatrix} \begin{bmatrix} C_\theta & 0 & -S_\theta \\ 0 & 1 & 0 \\ S_\theta & 0 & C_\theta \end{bmatrix} \begin{bmatrix} C_\psi & S_\psi & 0 \\ -S_\psi & C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} =$$

$$\begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & S_\phi C_\theta \\ C_\phi S_\theta C_\psi + S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\phi C_\theta \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \quad (1)$$

Since, Newton's second law is to be written in terms of the non-inertial coordinate system, the gravitational vector must be expressed in non-inertial coordinates. Applying Eq. (1) to the gravitational vector produces

$$\begin{Bmatrix} g_{x_b} \\ g_{y_b} \\ g_{z_b} \end{Bmatrix} = \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & S_\phi C_\theta \\ C_\phi S_\theta C_\psi + S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\phi C_\theta \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ g \end{Bmatrix} \quad (2)$$

$$= g \begin{Bmatrix} -S_\theta \\ S_\phi C_\theta \\ C_\phi C_\theta \end{Bmatrix}$$

The inverse of the transformation matrix in Eq. (1) is its transpose, making

$$\begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} = \begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix} \begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} \quad (3)$$

and from Eq. (3), the ground speed is related to the airspeed by

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{Bmatrix} V_{wx} \\ V_{wy} \\ V_{wz} \end{Bmatrix} + \begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \quad (4)$$

where the dot over a variable indicates the usual derivative with respect to time. The two vectors on the right-hand side of Eq. (4) are respectively the wind vector written in earth-fixed coordinates,

$$\bar{\mathbf{V}}_w = V_{wx}\bar{\mathbf{i}}_x + V_{wy}\bar{\mathbf{i}}_y + V_{wz}\bar{\mathbf{i}}_y \quad (5)$$

and the airspeed vector, which is written in body-fixed coordinates,

$$\bar{\mathbf{V}} = u\bar{\mathbf{i}}_{x_b} + v\bar{\mathbf{i}}_{y_b} + w\bar{\mathbf{i}}_{z_b} \quad (6)$$

The relationship between the non-inertial angular rates and the time rate of change of the Euler angles is[23,29]

$$\begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} = \begin{bmatrix} 1 & S_\phi S_\theta / C_\theta & C_\phi S_\theta / C_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi / C_\theta & C_\phi / C_\theta \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} \quad (7)$$

where the vector on the far right-hand side of Eq. (7) is the angular velocity vector written in body-fixed coordinates,

$$\bar{\omega} = p\bar{\mathbf{i}}_{x_b} + q\bar{\mathbf{i}}_{y_b} + r\bar{\mathbf{i}}_{z_b} \quad (8)$$

Equations (4) and (7) together provide the kinematic transformation equations in terms of Euler angles, which allow us to update the position and orientation of the aircraft with time. The gimbal lock singularity is seen in the last two terms of the first and last rows of Eq. (7). When the Euler elevation angle, θ, is ±90° these four terms go to infinity and the Euler angle integration becomes indeterminate. Combining Euler angle sequences can sometimes yield a singularity that is difficult to observe from a cursory inspection of the equations. Junkins and Shuster[50] propose a scheme involving spherical trigonometric relations to provide a clearer picture of the singularities. Alternate spacecraft-attitude dynamics models have been proposed which eliminate the Euler angle singularity,[51,52] although the aircraft community has not adopted them.

The Direction Cosine Formulation

The *direction-cosine matrix* can be formed from any of the symmetric or asymmetric Euler angles sets.[53] For example, the matrix on the right side of Eq. (1) is a direction-cosine matrix. One way to avoid the singularity in Eq. (7) is to treat the nine components of this matrix as a fundamental description of orientation. The elements of this matrix are called the direction cosines. If these nine direction cosines are known, the components of an arbitrary vector in body-fixed coordinates are quite simply related to the components of the same vector in inertial coordinates through the definition of the direction-cosine matrix, [C],

$$\begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} \equiv \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \tag{9}$$

and the gravitational vector, expressed in body-fixed coordinates, is

$$\begin{Bmatrix} g_{x_b} \\ g_{y_b} \\ g_{z_b} \end{Bmatrix} = g \begin{Bmatrix} C_{13} \\ C_{23} \\ C_{33} \end{Bmatrix} \tag{10}$$

The inverse of the direction-cosine matrix is simply its transpose, so the ground speed is related to the airspeed by

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{Bmatrix} V_{wx} \\ V_{wy} \\ V_{wz} \end{Bmatrix} + \begin{bmatrix} C_{11} & C_{21} & C_{31} \\ C_{12} & C_{22} & C_{32} \\ C_{13} & C_{23} & C_{33} \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{11}$$

When the body-fixed reference frame is undergoing rotation, the elements of the direction-cosine matrix are functions of time. To complete the kinematic formulation in terms of direction cosines, a set of differential equations relating the temporal derivatives of the nine elements of the direction-cosine matrix to the body-fixed angular rates is required. These nine equations, known as Poisson's kinematic equations, can be written in matrix form as[52,53]

$$\begin{bmatrix} \dot{C}_{11} & \dot{C}_{12} & \dot{C}_{13} \\ \dot{C}_{21} & \dot{C}_{22} & \dot{C}_{23} \\ \dot{C}_{31} & \dot{C}_{32} & \dot{C}_{33} \end{bmatrix} = \begin{bmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{bmatrix} \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} \tag{12}$$

It should be noted that there is no standard convention for numbering the direction cosines. Here, the numbering in Eq. (9) has been defined relative to the forward transformation, from earth-fixed coordinates to body-fixed coordinates. This is the most commonly used notation.[18,54-56] However, other authors have numbered the direction cosines relative to the inverse transformation, from body-fixed coordinates to earth-fixed coordinates.[33] One notation is simply the transpose of the other. Nevertheless, the reader should be careful to observe which notation is being used when reviewing a publication or simulation code.

There are only three degrees of freedom associated with orientation. However, there are nine components in the direction-cosine matrix. Thus, these nine components cannot be independent. There must be six levels of redundancy associated with this formulation. It can be shown that rigid-body rotation is a *proper orthogonal* transformation.[57] Therefore, the direction-cosine matrix is proper orthogonal and the inverse of the direction-cosine matrix must be its transpose,[56,57] $[\mathbf{C}]^{-1} = [\mathbf{C}]^T$. This requires the constraints

$$\begin{aligned} C_{11}^2 + C_{21}^2 + C_{31}^2 &= 1, \\ C_{12}^2 + C_{22}^2 + C_{32}^2 &= 1, \\ C_{13}^2 + C_{23}^2 + C_{33}^2 &= 1, \\ C_{11}C_{12} + C_{21}C_{22} + C_{31}C_{32} &= 0, \\ C_{11}C_{13} + C_{21}C_{23} + C_{31}C_{33} &= 0, \\ C_{12}C_{13} + C_{22}C_{23} + C_{32}C_{33} &= 0 \end{aligned} \tag{13}$$

These six constraints are usually called the redundancy relations. Mathematically, Eq. (12) preserves the orthogonality of the direction-cosine matrix. However, errors associated with integrating Eq. (12) numerically can cause degradation in the orthogonality of the matrix. Furthermore, these orthogonality errors can build up with time during the simulation. A method for avoiding this error buildup with the direction cosine formulation was first developed by Corbett and Wright[59] and is still commonly used today.

The direction cosine formulation contains no singularities and is frequently used by the aeronautics community[58] to avoid the possibility of gimbal lock. However, numerical integration of Eq. (12) is excessively time-consuming and a severe computational penalty is paid for its use.

The Euler Axis Formulation

The orientation of the non-inertial reference frame relative to the inertial reference frame can be described in terms of a single rotation through an angle, Θ, about a particular axis, $\bar{\mathbf{E}}$, which is commonly called the

Euler axis or the *eigenaxis*.[59] A comparison between this Euler axis rotation and the Euler angle rotations that are commonly used by the aircraft community is shown in Fig. 1.

The Euler axis rotation gives rise to a four component description of orientation. The four components in this description are the total rotation angle, Θ, and the three components of a vector directed along the Euler axis, E_x, E_y, and E_z. Since these four parameters describe an orientation having only three degrees of freedom, there must be some redundancy in the Euler axis description as well. In fact, by describing orientation in this manner, a fourth mathematical degree of freedom has been introduced. Clearly, the vector describing the orientation of the Euler axis could be of any arbitrary length. To remove this additional mathematical degree of freedom, a constraint must be applied, fixing the magnitude of the Euler axis vector. While this constraint is somewhat arbitrary, the usual and most obvious solution is to require the Euler axis vector to be of unit magnitude,

$$E_x^2 + E_y^2 + E_z^2 \equiv 1 \tag{14}$$

During the Euler axis rotation the orientation of the Euler axis is invariant. Thus, the vector, $\bar{\mathbf{E}}$, directed along the Euler axis has the same components in both inertial and non-inertial coordinates,

$$\begin{Bmatrix} E_x \\ E_y \\ E_z \end{Bmatrix} = \begin{Bmatrix} E_{x_b} \\ E_{y_b} \\ E_{z_b} \end{Bmatrix} \tag{15}$$

The components of an arbitrary vector, $\bar{v}$, in body-fixed coordinates are related to the components of the same vector in eath-fixed coordinates through what is commonly called Euler's formula[60]

$$\begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} = \begin{bmatrix} E_{xx}+C_\Theta & E_{xy}+E_zS_\Theta & E_{xz}-E_yS_\Theta \\ E_{xy}-E_zS_\Theta & E_{yy}+C_\Theta & E_{yz}+E_xS_\Theta \\ E_{xz}+E_yS_\Theta & E_{yz}-E_xS_\Theta & E_{zz}+C_\Theta \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \tag{16}$$

where $E_{ij} = E_iE_j(1-C_\Theta)$. Since the early work of Euler, a number of papers have been written on the derivation of equation (16).[61-67]

From Eq. (16) the gravitational vector expressed in non-inertial coordinates is

$$\begin{Bmatrix} g_{x_b} \\ g_{y_b} \\ g_{z_b} \end{Bmatrix} = g \begin{Bmatrix} E_{xz}-E_yS_\Theta \\ E_{yz}+E_xS_\Theta \\ E_{zz}+C_\Theta \end{Bmatrix} \tag{17}$$

The inverse of the transformation matrix in Eq. (16) is obtained by simply rotating through the negative of the total rotation angle that is used in the forward transformation,[54]

$$\begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} = \begin{bmatrix} E_{xx}+C_\Theta & E_{xy}-E_zS_\Theta & E_{xz}+E_yS_\Theta \\ E_{xy}+E_zS_\Theta & E_{yy}+C_\Theta & E_{yz}-E_xS_\Theta \\ E_{xz}-E_yS_\Theta & E_{yz}+E_xS_\Theta & E_{zz}+C_\Theta \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \tag{18}$$

Thus, the ground speed can be related to the airspeed through

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{Bmatrix} V_{wx} \\ V_{wy} \\ V_{wz} \end{Bmatrix} + \begin{bmatrix} E_{xx}+C_\Theta & E_{xy}-E_zS_\Theta & E_{xz}+E_yS_\Theta \\ E_{xy}+E_zS_\Theta & E_{yy}+C_\Theta & E_{yz}-E_xS_\Theta \\ E_{xz}-E_yS_\Theta & E_{yz}+E_xS_\Theta & E_{zz}+C_\Theta \end{bmatrix} \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{19}$$

The relationship between the non-inertial angular rates and the time rate of change of the Euler axis rotation parameters is given by

$$\begin{Bmatrix} \dot{\Theta} \\ \dot{E}_x \\ \dot{E}_y \\ \dot{E}_z \end{Bmatrix} = \frac{1}{2} \begin{bmatrix} 2E_x & 2E_y & 2E_z \\ E'_{xx}+C/S & E'_{xy}-E_z & E'_{xz}+E_y \\ E'_{xy}+E_z & E'_{yy}+C/S & E'_{yz}-E_x \\ E'_{xz}-E_y & E'_{yz}+E_x & E'_{zz}+C/S \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} \tag{20}$$

where $E'_{ij} = -E_iE_j\,C/S$, $S=\sin(\Theta/2)$, and $C=\cos(\Theta/2)$. Equations (19) and (20) are the kinematic transformation equations in terms of the Euler axis rotation parameters. Notice that Eq. (20) also has a singularity. When the total rotation angle, Θ, is zero, the integration of these equations is indeterminate.

The Euler-Rodrigues Quaternion Formulation

The Euler-Rodrigues formulation is related to the Euler axis formulation through a simple change of variables. The four parameters in the Euler axis

description of orientation are used to define four different parameters, which are somewhat more convenient. These four new parameters are defined as

$$\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} \equiv \begin{Bmatrix} \cos(\Theta/2) \\ E_x \sin(\Theta/2) \\ E_y \sin(\Theta/2) \\ E_z \sin(\Theta/2) \end{Bmatrix} \tag{21}$$

These four parameters are known as the *Euler-Rodrigues symmetric parameters*[68,69] or the quaternion of finite rotation.[70] The Euler-Rodrigues formulation for rigid body kinematics was developed well before 1844, when the well-known mathematician William R. Hamilton first developed the quaternion and the detailed theory of a noncommutative algebraic system known as quaternion algebra.[71-73] Much has been written about the life of Hamilton[74-76] and his moment of enlightenment concerning quaternion formulas while crossing a bridge with his wife. Hamilton, in a letter to his son, noted he could not resist the impulse to carve the fundamental quaternion formula into the stone on the bridge. However, it is clear that Hamilton did not develop quaternion algebra as means of describing rotational transformations.

The brief review of the history of the Euler-Rodrigues formulation that is presented here is taken primarily from Altmann,[77,78] Shuster,[18] and Cheng and Gupta.[79] In 1758 an early work of Euler[80] showed that any differential movement of a rigid body can be expressed as a translation and a rotation about some specific axis. Euler's theorem on the motion of a rigid body,[59] as well as Euler's formula[60] were both published in 1775. The first publication of the derivation of the Euler angles[81] was in 1862. Although the work was published posthumously, the date is probably in error as Euler died in 1783.

Whether Euler knew of the Euler-Rodrigues symmetric parameters is a subject of debate. In 1770, Euler[68] developed four symmetric parameters for orthogonal transformations (without the use of half angles). Roberson[82] and Jacobi[83] argue that Euler had presented a rotation matrix in terms of the so-called Euler-Rodrigues parameters. Shuster[18] notes that the symmetric parameters developed by Euler in the 1770 paper contained sign errors and formed an improper orthogonal matrix. It is important to note that Euler viewed a matrix as a table. The matrix as mathematical object would not evolve until vector space was studied by Grassman[84,85], Gibbs[86] and Heaviside.[87] As a side note, the chief properties of vectors were worked out in Hamilton's investigation of quaternion algebra.[88]

In 1840, four years before Hamilton began his algebraic study of quaternions, Olinde Rodrigues[69] published his work on the Euler-Rodrigues symmetric parameters, the rules for the compositions, and a geometrical construction for combining two rotations. Unlike Hamilton, whose accomplishments are well documented, the only published factual article on Rodrigues did not appear until 1980.[89] Earlier historians have invented a collaborator of Rodrigues by the name of Olinde[90,91] or have mistaken his name as Rodrigue[92] or Rodriques.[93] Additional historical details can be found in Kline,[94] Van der Waerden,[95] Crowe,[96] and McDuffee.[97]

There is no universal agreement on the choice of indices for the Euler-Rodrigues symmetric parameters. Most authors use the indices 1 through 4 or 0 through 3. Some authors have chosen indices 1 through 3 for the vector components while using 4 as the scalar index,[18,98,99] whereas others have used 1 for the scalar index and 2 through 4 for the vector component.[100] The scalar index has also been chosen as 0 with indices 1 through 3 denoting the vector components.[15,17,38,101-106] There is not even universal agreement on the order of the vector components. Whereas most authors assign the x- y- z-components in ascending order, at least two authors have used 4 to denote the x-component, 3 for the y-component, and 2 for the z-component.[34,37] To avoid confusion in the present paper, the vector components are explicitly labeled using the subscripts x, y, and z while a 0 subscript is used to denote the scalar component.

Since the four parameters defined by Eq. (21) uniquely describe an orientation having only three degrees of freedom, these four parameters must be related in some way. This relation is easily seen by squaring the four components of Eq. (21) and adding them together. This gives

$$\begin{aligned} &e_0^2 + e_x^2 + e_y^2 + e_z^2 = \\ &\cos^2(\Theta/2) + (E_x^2 + E_y^2 + E_z^2)\sin^2(\Theta/2) \end{aligned} \tag{22}$$

Since the Euler axis vector, $\bar{\mathbf{E}}$, is a unit vector and $\cos^2(\chi) + \sin^2(\chi) = 1$, it follows that

$$e_0^2 + e_x^2 + e_y^2 + e_z^2 = 1 \tag{23}$$

The transformation in Eq. (16) can be written in terms of half the total rotation angle by applying the trigonometric identities, $\sin(\chi) = 2\sin(\chi/2)\cos(\chi/2)$, $\cos(x) = \cos^2(\chi/2) - \sin^2(\chi/2)$ and $1 - \cos(\chi) = 2\sin^2(\chi/2)$. Thus, using the notation $S = \sin(\Theta/2)$, $C = \cos(\Theta/2)$, and $E_{ij} = 2E_iE_jS^2$, Eq. (16) can be rewritten as

$$\begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} =$$

$$\begin{bmatrix} E_{xx}+C^2-S^2 & E_{xy}+2E_zSC & E_{xz}-2E_ySC \\ E_{xy}-2E_zSC & E_{yy}+C^2-S^2 & E_{yz}+2E_xSC \\ E_{xz}+2E_ySC & E_{yz}-2E_xSC & E_{zz}+C^2-S^2 \end{bmatrix} \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \tag{24}$$

Now, applying Eq. (21) and recognizing that $E_{ij} = 2e_ie_j$ and $S^2 = (E_x^2 + E_y^2 + E_z^2)S^2 = e_x^2 + e_y^2 + e_z^2$, Eq. (24) becomes

$$\begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} =$$

$$\begin{bmatrix} e_x^2+e_0^2-e_y^2-e_z^2 & 2(e_xe_y+e_ze_0) & 2(e_xe_z-e_ye_0) \\ 2(e_xe_y-e_ze_0) & e_y^2+e_0^2-e_x^2-e_z^2 & 2(e_ye_z+e_xe_0) \\ 2(e_xe_z+e_ye_0) & 2(e_ye_z-e_xe_0) & e_z^2+e_0^2-e_x^2-e_y^2 \end{bmatrix} \times \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} \tag{25}$$

The transformation equation given by Eq. (25) can be used to obtain the gravitational acceleration vector in body-fixed coordinates,

$$\begin{Bmatrix} g_{x_b} \\ g_{y_b} \\ g_{z_b} \end{Bmatrix} = g \begin{Bmatrix} 2(e_xe_z-e_ye_0) \\ 2(e_ye_z+e_xe_0) \\ e_z^2+e_0^2-e_x^2-e_y^2 \end{Bmatrix} \tag{26}$$

The inverse of the transformation matrix in Eq. (25) is

$$\begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix} =$$

$$\begin{bmatrix} e_x^2+e_0^2-e_y^2-e_z^2 & 2(e_xe_y-e_ze_0) & 2(e_xe_z+e_ye_0) \\ 2(e_xe_y+e_ze_0) & e_y^2+e_0^2-e_x^2-e_z^2 & 2(e_ye_z-e_xe_0) \\ 2(e_xe_z-e_ye_0) & 2(e_ye_z+e_xe_0) & e_z^2+e_0^2-e_x^2-e_y^2 \end{bmatrix} \times \begin{Bmatrix} v_{x_b} \\ v_{y_b} \\ v_{z_b} \end{Bmatrix} \tag{27}$$

So the ground speed is related to the airspeed by

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{Bmatrix} V_{wx} \\ V_{wy} \\ V_{wz} \end{Bmatrix} +$$

$$\begin{bmatrix} e_x^2+e_0^2-e_y^2-e_z^2 & 2(e_xe_y-e_ze_0) & 2(e_xe_z+e_ye_0) \\ 2(e_xe_y+e_ze_0) & e_y^2+e_0^2-e_x^2-e_z^2 & 2(e_ye_z-e_xe_0) \\ 2(e_xe_z-e_ye_0) & 2(e_ye_z+e_xe_0) & e_z^2+e_0^2-e_x^2-e_y^2 \end{bmatrix} \times \begin{Bmatrix} u \\ v \\ w \end{Bmatrix} \tag{28}$$

The time rate of change of the Euler-Rodrigues symmetric parameters can be related to the time rate of change of the Euler axis rotation parameters. Differentiating Eq. (21), produces

$$\begin{Bmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{Bmatrix} = \begin{Bmatrix} -\sin(\Theta/2) \\ E_x\cos(\Theta/2) \\ E_y\cos(\Theta/2) \\ E_z\cos(\Theta/2) \end{Bmatrix} \frac{\dot{\Theta}}{2} + \begin{Bmatrix} 0 \\ \dot{E}_x\sin(\Theta/2) \\ \dot{E}_y\sin(\Theta/2) \\ \dot{E}_z\sin(\Theta/2) \end{Bmatrix} \tag{29}$$

Using Eq. (20) to express the time rate of change of the Euler axis rotation parameters in terms of the non-inertial angular rates, Eq. (29) can be written as

$$\begin{Bmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{Bmatrix} = \frac{1}{2} \begin{bmatrix} -E_xS & -E_yS & -E_zS \\ C & -E_zS & E_yS \\ E_zS & C & -E_xS \\ -E_yS & E_xS & C \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} \tag{30}$$

or after applying Eq. (21)

$$\begin{Bmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{Bmatrix} = \frac{1}{2} \begin{bmatrix} -e_x & -e_y & -e_z \\ e_0 & -e_z & e_y \\ e_z & e_0 & -e_x \\ -e_y & e_x & e_0 \end{bmatrix} \begin{Bmatrix} p \\ q \\ r \end{Bmatrix} \tag{31}$$

Since Eq. (31) is linear in both the non-inertial angular rates and the Euler-Rodrigues symmetric parameters, it can as well be written as[105,106]

$$\begin{Bmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{Bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} \tag{32}$$

Equations (28) and (31) or (32) provide the kinematic

transformation equations in terms of the Euler-Rodrigues symmetric parameters.

Robinson[37] detailed the advantages of the Euler-Rodrigues quaternion formulation over the Euler angle or the direction cosine formulation almost fifty years ago. Most of his observations, obtained on analog computers, are still valid in the digital world. When the Euler elevation angle, θ, is $\pm 90^{\circ}$, the Euler angle integration becomes indeterminate. Despite the singularity, the Euler angle formulation is widely used because the three Euler angles have the simple interpretation of heading, elevation angle, and bank angle.[109,110] The Euler-Rodrigues formulation is free of singularities, however, the physical interpretation of the quaternion is much less intuitive than that associated with the Euler angles. In a mathematical study, Stuelpnagel[100] considered parameterization of a general three-parameter rotation group, four-parameter rotation group, and five- and six-parameter groups. He proves that the three-parameter rotation group leads to nonlinear kinematic equations and that the Euler-Rodrigues symmetric parameters represent the smallest number of parameters (four) with linear kinematic equations. Errors associated with numerical integration of the kinematic equations for attitude have been characterized for both the quaternion and the direction cosine parameterizations and the superiority of the quaternion parameterization is well documented.[104,110-115] Lovren and Piper[116] show that, for the isolated case of classical coning motion, integration errors are similar for the direction-cosine parameterization and the quaternion parameterization. Another advantage of the Euler-Rodrigues formulation is the application of Kalman filtering to quaternion estimation.[117-120] For the reasons stated above, the spacecraft community often utilizes the Euler-Rodrigues formulation for attitude control systems.[105,121-140]

Perhaps the greatest advantage of the quaternion formulation over either the Euler angle formulation or the direction-cosine formulation is increased computational speed. Numerical integration of the nine component direction-cosine formulation is excessively time-consuming when compared with the four component Euler-Rodrigues formulation. The trigonometric functions in the Euler angle transformation matrix make simulations that use this nonlinear formulation much more computationally intensive than those using the linear equations of the quaternion formulation. Furthermore, the computational advantage of the quaternion formulation can be extended even further through use of Hamilton's quaternion algebra. While the Euler-Rodrigues formulation was originally developed before quaternion algebra was conceived, the most computationally efficient algorithms are in fact based on this mathematical rule set, first introduced by Hamilton.[71] To demonstrate this computational advantage, a brief introduction to quaternion algebra is presented here.

Quaternion Algebra

A general quaternion, $\mathbf{Q}$, is defined as

$$\mathbf{Q} \equiv Q_0 + Q_x \bar{\mathbf{i}}_x + Q_y \bar{\mathbf{i}}_y + Q_z \bar{\mathbf{i}}_z \tag{33}$$

where Q_0, Q_x, Q_y, and Q_z are scalars and $\bar{\mathbf{i}}_x$, $\bar{\mathbf{i}}_y$, and $\bar{\mathbf{i}}_z$ are unit vectors in the Cartesian x-, y-, and z-directions respectively. A quaternion has both vector and scalar properties. Vectors and scalars can be thought of as special cases of the more general quaternion. A scalar is a quaternion with Q_x, Q_y, and Q_z equal to zero and a vector is a quaternion where Q_o equals zero.

For quaternion multiplication, a procedure called the *quaternion product* is defined. Since a vector is a special case of a quaternion, care should be taken to distinguish the quaternion product from either the dot product or the cross product. Here, the quaternion product will be indicated using the operator, $\otimes$. The quaternion products of the usual Cartesian unit vectors are defined according to the following rule set:

$$\begin{aligned}
&\bar{\mathbf{i}}_x \otimes \bar{\mathbf{i}}_x \equiv -1, \quad \bar{\mathbf{i}}_x \otimes \bar{\mathbf{i}}_y \equiv \bar{\mathbf{i}}_z, \quad \bar{\mathbf{i}}_x \otimes \bar{\mathbf{i}}_z \equiv -\bar{\mathbf{i}}_y \\
&\bar{\mathbf{i}}_y \otimes \bar{\mathbf{i}}_x \equiv -\bar{\mathbf{i}}_z, \quad \bar{\mathbf{i}}_y \otimes \bar{\mathbf{i}}_y \equiv -1, \quad \bar{\mathbf{i}}_y \otimes \bar{\mathbf{i}}_z \equiv \bar{\mathbf{i}}_x, \\
&\bar{\mathbf{i}}_z \otimes \bar{\mathbf{i}}_x \equiv \bar{\mathbf{i}}_y, \quad \bar{\mathbf{i}}_z \otimes \bar{\mathbf{i}}_y \equiv -\bar{\mathbf{i}}_x, \quad \bar{\mathbf{i}}_z \otimes \bar{\mathbf{i}}_z \equiv -1
\end{aligned} \tag{34}$$

The quaternion product simply follows the distributive law. Thus, the quaternion product of one quaternion, $\mathbf{A}$, with another quaternion, $\mathbf{B}$, is[141]

$$\begin{aligned}
\mathbf{A} \otimes \mathbf{B} &= (A_0 + A_x \bar{\mathbf{i}}_x + A_y \bar{\mathbf{i}}_y + A_z \bar{\mathbf{i}}_z) \\
&\quad \otimes (B_0 + B_x \bar{\mathbf{i}}_x + B_y \bar{\mathbf{i}}_y + B_z \bar{\mathbf{i}}_z) \\
&= (A_0 B_0 - A_x B_x - A_y B_y - A_z B_z) \\
&\quad + (A_0 B_x + A_x B_0 + A_y B_z - A_z B_y) \bar{\mathbf{i}}_x \\
&\quad + (A_0 B_y - A_x B_z + A_y B_0 + A_z B_x) \bar{\mathbf{i}}_y \\
&\quad + (A_0 B_z + A_x B_y - A_y B_x + A_z B_0) \bar{\mathbf{i}}_z
\end{aligned} \tag{35}$$

The quaternion products defined in Eq. (34) are nearly like cross products, following the right hand rule. However, special treatment is given to the quaternion product of a unit vector with itself. It should also be noted that, in general, $\mathbf{A} \otimes \mathbf{B} \neq \mathbf{B} \otimes \mathbf{A}$ so the quaternion product is not commutative. Also notice that the quaternion product reduces to simple multiplication for the special case when either operand is a scalar. However, it does not reduce to either the dot product or the cross product when both operands are vectors.

In general, the quaternion product of two simple vectors is a four-component quaternion, equal to the negative of the dot product added to the cross product,

$$\begin{aligned}\bar{\mathbf{A}}\otimes\bar{\mathbf{B}} &= -(A_xB_x + A_yB_y + A_zB_z)\\ &+(A_yB_z - A_zB_y)\,\bar{\mathbf{i}}_x\\ &+(A_zB_x - A_xB_z)\,\bar{\mathbf{i}}_y\\ &+(A_xB_y - A_yB_x)\,\bar{\mathbf{i}}_z \;=\; -\bar{\mathbf{A}}\cdot\bar{\mathbf{B}} + \bar{\mathbf{A}}\times\bar{\mathbf{B}}\end{aligned} \tag{36}$$

Quaternions not only have properties of scalars and vectors, but quaternion algebra also has similarities to complex algebra. The magnitude of a quaternion is defined similar to that of a complex number or a vector,

$$|\mathbf{Q}| \equiv \sqrt{Q_0^2 + Q_x^2 + Q_y^2 + Q_z^2} \tag{37}$$

Also, a conjugate of a quaternion is defined as,

$$\mathbf{Q}^* \equiv Q_0 - Q_x\bar{\mathbf{i}}_x - Q_y\bar{\mathbf{i}}_y - Q_z\bar{\mathbf{i}}_z \tag{38}$$

where the asterisk indicates a quaternion conjugate. Using Eq. (38) with Eq. (35) produces

$$\mathbf{Q}\otimes\mathbf{Q}^* \;=\; Q_0^2 + Q_x^2 + Q_y^2 + Q_z^2 \;=\; |\mathbf{Q}|^2 \tag{39}$$

Thus, similar to a complex variable, the quaternion product of a quaternion with its conjugate generates a scalar equal to the square of the magnitude of the quaternion.

The four Euler-Rodrigues symmetric parameters can be thought of as the four components of a particular unit quaternion, $\mathbf{e}$, having components (e_0, e_x, e_y, e_z), which uniquely specifies the orientation of the non-inertial coordinate system relative to the inertial coordinate system. Furthermore, the rotational transformation given by Eq. (25) can also be expressed in terms of quaternion products. Let $\bar{\nu}$ be an arbitrary vector having quaternion components $(0, \nu_{x_b}, \nu_{y_b}, \nu_{z_b})$ in the non-inertial coordinate system and having quaternion components $(0, \nu_x, \nu_y, \nu_z)$ in the inertial coordinate system. The components of $\bar{\nu}$ in non-inertial coordinates are related to the components of $\bar{\nu}$ in inertial coordinates through the quaternion transformation[17]

$$\bar{\nu}_b \;=\; \mathbf{e}^*\otimes\left(\bar{\nu}\otimes\mathbf{e}\right) \tag{40}$$

Expanding Eq. (40) using Eq. (35), the orthogonal transformation can be written as a two step process using a temporary quaternion, $\mathbf{T}$,

$$\begin{aligned}\mathbf{T} \;=\; \bar{\nu}\otimes\mathbf{e} \;=\;& (-\nu_xe_x - \nu_ye_y - \nu_ze_z)\\ &+(\nu_xe_0 + \nu_ye_z - \nu_ze_y)\,\bar{\mathbf{i}}_x\\ &+(-\nu_xe_z + \nu_ye_0 + \nu_ze_x)\,\bar{\mathbf{i}}_y\\ &+(\nu_xe_y - \nu_ye_x + \nu_ze_0)\,\bar{\mathbf{i}}_z\end{aligned} \tag{41}$$

$$\begin{aligned}\bar{\nu}_b \;=\; \mathbf{e}^*\otimes\mathbf{T} \;=\;& (e_0T_x - e_xT_0 - e_yT_z + e_zT_y)\,\bar{\mathbf{i}}_{x_b}\\ &+(e_0T_y + e_xT_z - e_yT_0 - e_zT_x)\,\bar{\mathbf{i}}_{y_b}\\ &+(e_0T_z - e_xT_y + e_yT_x - e_zT_0)\,\bar{\mathbf{i}}_{z_b}\end{aligned} \tag{42}$$

The advantage of Eqs. (41) and (42) over Eq. (25) is a matter of reduced computation time. The rotational transformation as expressed in Eq. (25) requires 39 multiplications and 21 additions. On the other hand, computing the same transformation from Eqs. (41) and (42) requires only 24 multiplications and 17 additions. This translates to a significant computational saving. For typical modern computers, the transformation computed from Eq. (25) requires about 50 percent greater computation time than does that computed from Eqs. (41) and (42). This can be quite significant in a flight simulator, which requires a very large number of such transformations for the visual display.[142]

For aircraft that are not all-attitude vehicles, such as transports, helicopters, and roll-stabilized missiles, Euler angles are often used to compute attitude because the singularity is not encountered.[143] However, the computational advantage of the quaternion transformation is even more impressive when compared with the Euler angle transformation given by Eq. (1). Even when the trigonometric functions in Eq. (1) are evaluated only once, the transformation computed from Eq. (1) requires about 11 times as long to evaluate as the quaternion transformation computed from Eqs. (41) and (42). Thus, even ignoring the singularity in the Euler angle formulation, the quaternion formulation is far superior to the Euler angle formulation, based on computational efficiency alone.

In the mid-1980s, algorithms were developed, which further reduced the computation time for quaternion multiplication on the hardware available at that time. For example, Dvornychenko,[144] based on the work of Winograd,[145,146] presented two algorithms that were claimed to reduce computation time. The first of these algorithms requires 11 multiplications and 19 additions. The second requires 10 multiplications and 26 additions. This compares to 16 multiplications and 12 additions for the conventional quaternion multiplication described in Eq. (35). The hardware available at that time required something on the order of 5 instruction

cycles for multiplication and only 1 cycle for addition. Thus, trading 5 multiplications for 7 additions would result in a significant reduction in computation time, for the hardware of that era. However, typical modern hardware requires only 1 instruction cycle for multiplication and 1 instruction cycle for addition. Thus, the algorithms that were developed in the 1980s to speed up quaternion transformations will actually slow down the transformations when used with today's hardware. As late as 1993, Schuter[18] has claimed that these algorithms will speed up the transformations. For simulations run on modern hardware, these old algorithms should be replaced with the more straightforward algorithm specified by Eq. (35).

The system of differential equation, which governs the change in the transformation quaternion with time, can also be written in terms of a quaternion product.[99,147] This differential system, specified by either Eq. (31) or Eq. (32), can be written as

$$\dot{\mathbf{e}} = \tfrac{1}{2}\mathbf{e}\otimes\bar{\omega} \tag{43}$$

where $\bar{\omega}$ is the angular velocity vector in body-fixed coordinates, as defined by Eq. (8). Writing the quaternion differential equation in this form provides no computational saving over the matrix form given by Eq. (32). From Eq. (35), it is observed that the quaternion product in Eq. (43) requires 16 multiplications and 12 additions. This is exactly equivalent to that required for the matrix multiplication in Eq. (32). However, the matrix multiplication in Eq. (31) requires only 12 multiplications and 8 additions. Thus, the temporal derivative of the transformation quaternion should always be computed from Eq. (31).

Relationships between the Quaternion and other Attitude Descriptors

The physical interpretation of the Euler-Rodrigues quaternion is much less intuitive than that associated with the Euler angles. For this reason it is convenient to be able to relate the quaternion to the Euler angles. Such a relation can be obtained by combining Eqs. (1) and (25). These two equations require

$$\begin{bmatrix} e_x^2+e_0^2-e_y^2-e_z^2 & 2(e_xe_y+e_ze_0) & 2(e_xe_z-e_ye_0) \\ 2(e_xe_y-e_ze_0) & e_y^2+e_0^2-e_x^2-e_z^2 & 2(e_ye_z+e_xe_0) \\ 2(e_xe_z+e_ye_0) & 2(e_ye_z-e_xe_0) & e_z^2+e_0^2-e_x^2-e_y^2 \end{bmatrix} = \begin{bmatrix} C_\theta C_\psi & C_\theta S_\psi & -S_\theta \\ S_\phi S_\theta C_\psi - C_\phi S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & S_\phi C_\theta \\ C_\phi S_\theta C_\psi + S_\phi S_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi & C_\phi C_\theta \end{bmatrix} \tag{44}$$

This matrix equation provides nine scalar equations relating the four components of the Euler-Rodrigues quaternion, $\mathbf{e}$, to the three Euler angles, ϕ, θ, and ψ. However, the nine components of the matrices on both sides of Eq. (44) each describe a rotation having only three degrees of freedom. Thus, there are six levels of redundancy in Eq. (44). Additionally, Eq. (23) must also be satisfied, which simply requires that $\mathbf{e}$ be a unit quaternion. The three degrees of freedom in Eq. (44) combined with the requirement in Eq. (23) provide exactly the four degrees of freedom needed to solve for the four components of the quaternion $\mathbf{e}$.

Combining the diagonal components of Eq. (44) with the requirement for a unit quaternion expressed in Eq. (23), a four by four system of algebraic equations for the squares of the quaternion components is obtained,

$$\begin{bmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{Bmatrix} e_0^2 \\ e_x^2 \\ e_y^2 \\ e_z^2 \end{Bmatrix} = \begin{Bmatrix} C_\theta C_\psi \\ S_\phi S_\theta S_\psi + C_\phi C_\psi \\ C_\phi C_\theta \\ 1 \end{Bmatrix} \tag{45}$$

This system is readily solved by elimination to give

$$\begin{Bmatrix} e_0^2 \\ e_x^2 \\ e_y^2 \\ e_z^2 \end{Bmatrix} = \frac{1}{4}\begin{Bmatrix} 1+C_\theta C_\psi + S_\phi S_\theta S_\psi + C_\phi C_\theta + C_\phi C_\psi \\ 1-C_\phi C_\theta - S_\phi S_\theta S_\psi - C_\phi C_\psi + C_\theta C_\psi \\ 1-C_\theta C_\psi - C_\phi C_\theta + S_\phi S_\theta S_\psi + C_\phi C_\psi \\ 1-C_\theta C_\psi + C_\phi C_\theta - S_\phi S_\theta S_\psi - C_\phi C_\psi \end{Bmatrix} \tag{46}$$

Applying the half angle identities, Eq. (46) is written as

$$\begin{Bmatrix} e_0^2 \\ e_x^2 \\ e_y^2 \\ e_z^2 \end{Bmatrix} = \begin{Bmatrix} (C_{\phi/2}C_{\theta/2}C_{\psi/2} + S_{\phi/2}S_{\theta/2}S_{\psi/2})^2 \\ (S_{\phi/2}C_{\theta/2}C_{\psi/2} - C_{\phi/2}S_{\theta/2}S_{\psi/2})^2 \\ (C_{\phi/2}S_{\theta/2}C_{\psi/2} + S_{\phi/2}C_{\theta/2}S_{\psi/2})^2 \\ (S_{\phi/2}S_{\theta/2}C_{\psi/2} - C_{\phi/2}C_{\theta/2}S_{\psi/2})^2 \end{Bmatrix} \tag{47}$$

Each of these component equations has two possible solutions so the signs are indeterminate at this point. The off-diagonal components of Eq. (44) may be extracted to produce

$$\begin{bmatrix} 0 & 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & -2 & 2 & 0 & 0 \\ 0 & -2 & 0 & 0 & 2 & 0 \\ 0 & 2 & 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 & 0 & 2 \\ -2 & 0 & 0 & 0 & 0 & 2 \end{bmatrix} \begin{Bmatrix} e_0e_x \\ e_0e_y \\ e_0e_z \\ e_xe_y \\ e_xe_z \\ e_ye_z \end{Bmatrix} = \begin{Bmatrix} C_\theta S_\psi \\ S_\phi S_\theta C_\psi - C_\phi S_\psi \\ -S_\theta \\ C_\phi S_\theta C_\psi + S_\phi S_\psi \\ S_\phi C_\theta \\ C_\phi S_\theta S_\psi - S_\phi C_\psi \end{Bmatrix} \tag{48}$$

This simple algebraic system is easily solved by adding and subtracting appropriate pairs of equations. Replacing the first and second equations with their sum and their difference and doing likewise with the other two pairs of equations results in

$$\begin{Bmatrix} e_0e_x \\ e_0e_y \\ e_0e_z \\ e_xe_y \\ e_xe_z \\ e_ye_z \end{Bmatrix} = \frac{1}{4}\begin{Bmatrix} S_\phi C_\theta - C_\phi S_\theta S_\psi + S_\phi C_\psi \\ C_\phi S_\theta C_\psi + S_\phi S_\psi + S_\theta \\ C_\theta S_\psi - S_\phi S_\theta C_\psi + C_\phi S_\psi \\ C_\theta S_\psi + S_\phi S_\theta C_\psi - C_\phi S_\psi \\ C_\phi S_\theta C_\psi + S_\phi S_\psi - S_\theta \\ S_\phi C_\theta + C_\phi S_\theta S_\psi - S_\phi C_\psi \end{Bmatrix} \tag{49}$$

Using Eq. (47) in Eq. (49) and applying the half angle identities, the off-diagonal components of Eq. (44) reduce to

$$\begin{Bmatrix} s_0s_x(S_\phi C_\theta - C_\phi S_\theta S_\psi + S_\phi C_\psi) \\ s_0s_y(C_\phi S_\theta C_\psi + S_\phi S_\psi + S_\theta) \\ s_0s_z(-C_\theta S_\psi + S_\phi S_\theta C_\psi - C_\phi S_\psi) \\ s_xs_y(C_\theta S_\psi + S_\phi S_\theta C_\psi - C_\phi S_\psi) \\ s_xs_z(-C_\phi S_\theta C_\psi - S_\phi S_\psi + S_\theta) \\ s_ys_z(-S_\phi C_\theta - C_\phi S_\theta S_\psi + S_\phi C_\psi) \end{Bmatrix} = \begin{Bmatrix} S_\phi C_\theta - C_\phi S_\theta S_\psi + S_\phi C_\psi \\ C_\phi S_\theta C_\psi + S_\phi S_\psi + S_\theta \\ C_\theta S_\psi - S_\phi S_\theta C_\psi + C_\phi S_\psi \\ C_\theta S_\psi + S_\phi S_\theta C_\psi - C_\phi S_\psi \\ C_\phi S_\theta C_\psi + S_\phi S_\psi - S_\theta \\ S_\phi C_\theta + C_\phi S_\theta S_\psi - S_\phi C_\psi \end{Bmatrix} \tag{50}$$

where s_0, s_x, s_y, and s_z are the unknown signs of the quaternion components from Eq. (47). Thus, the off-diagonal components of Eq. (44) provide only three additional pieces of information, $s_0s_x=1$, $s_0s_y=1$, and $s_0s_z=-1$. Using these signs with Eq. (47), only two possible solutions to Eq. (44) exist, which are

$$\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} = \pm\begin{Bmatrix} C_{\phi/2}C_{\theta/2}C_{\psi/2} + S_{\phi/2}S_{\theta/2}S_{\psi/2} \\ S_{\phi/2}C_{\theta/2}C_{\psi/2} - C_{\phi/2}S_{\theta/2}S_{\psi/2} \\ C_{\phi/2}S_{\theta/2}C_{\psi/2} + S_{\phi/2}C_{\theta/2}S_{\psi/2} \\ C_{\phi/2}C_{\theta/2}S_{\psi/2} - S_{\phi/2}S_{\theta/2}C_{\psi/2} \end{Bmatrix} \tag{51}$$

The positive sign in Eq. (51) corresponds to the right-hand Euler-Rodrigues quaternion, which is based on a right-hand rotation about the Euler axis. The negative sign results in a left-hand quaternion, which is not normally used.

The inverse of Eq. (51) is obtained from Eq. (44) as well. This is rather straightforward and yields[104]

$$\begin{Bmatrix} \phi \\ \theta \\ \psi \end{Bmatrix} = \begin{Bmatrix} \text{atan2}[2(e_0e_x + e_ye_z),(e_0^2 + e_z^2 - e_x^2 - e_y^2)] \\ \text{asin}[2(e_0e_y - e_xe_z)] \\ \text{atan2}[2(e_0e_z + e_xe_y),(e_0^2 + e_x^2 - e_y^2 - e_z^2)] \end{Bmatrix} \tag{52}$$

The function atan2 in Eq. (52) is a two-argument arctangent that returns a result in the proper quadrant, such as the atan2 intrinsic provided in Fortran and C. The two-argument function is not needed for the elevation angle, θ, because this angle is defined only in the range $-\pi/2$ to $\pi/2$.

The Euler-Rodrigues quaternion is also related to the total rotation angle and the Euler axis components through the definition of the Euler-Rodrigues symmetric parameters,

$$\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} \equiv \begin{Bmatrix} \cos(\Theta/2) \\ E_x \sin(\Theta/2) \\ E_y \sin(\Theta/2) \\ E_z \sin(\Theta/2) \end{Bmatrix} \tag{53}$$

The inverse of Eq. (53) is readily found to be

$$\begin{Bmatrix} \Theta \\ E_x \\ E_y \\ E_z \end{Bmatrix} = \begin{Bmatrix} 2\cos^{-1}(e_0) \\ e_x/\sin(\Theta/2) \\ e_y/\sin(\Theta/2) \\ e_z/\sin(\Theta/2) \end{Bmatrix} \tag{54}$$

Thus, from knowledge of the four components of the Euler-Rodrigues quaternion, the orientation of the non-inertial reference frame with respect to the inertial reference frame can be expressed as a single rotation about a known axis through a known angle. For non-zero rotation angles, the three vector components of this unit quaternion compose a vector directed along the Euler axis. The scalar component of this quaternion is equal to the cosine of one half the angle of rotation about this axis, in a direction defined by the right hand rule. If the scalar component of this quaternion is ± 1, the orientation of the Euler axis is indeterminate. However, a scalar component of ± 1 would indicate a total rotation angle of zero, making knowledge of the Euler axis unnecessary. It is important to recognize that, like the Euler axis vector, $\bar{\mathbf{E}}$, the four components of the unit quaternion, $\mathbf{e}$, are the same in both the inertial and the non-inertial reference frames. Thus, a reference frame need not be specified when referring to the components of this particular quaternion.

During the 1970's the spacecraft community utilized both the direction cosines and the Euler-Rodrigues quaternion to describe attitude. For example, in space shuttle steering algorithms the command attitude was computed as a direction-cosine matrix, whereas the attitude error was more conveniently described as a quaternion.[101] The quaternion was used to describe the attitude error because of its simple physical interpretation. The error axis coincides with the vector part of the error quaternion and the total error is readily determined from the scalar part. To compute the error quaternion, the command-attitude quaternion was first extracted from the direction-cosine matrix. Several other space shuttle flight algorithms also required the determination of an attitude quaternion from a direction-cosines matrix. Consequently, a number of papers have been written on extracting the quaternion from the direction cosines.[102,148-151] The most efficient method was first published in 1978 by Shepperd[103] and has been restated by Shuster and Natanson.[98] The aircraft and missile community is still using both the direction cosines[58,152] and the Euler-Rodrigues quaternion[35,153-155] as attitude descriptors for aircraft flight simulation.

The direction cosines can be expressed in terms of the four components of the Euler-Rodrigues quaternion by equating the left-hand side of Eqs. (9) to the left-hand side of Eqs. (25).

$$\begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} = \begin{bmatrix} e_x^2+e_0^2-e_y^2-e_z^2 & 2(e_xe_y+e_ze_0) & 2(e_xe_z-e_ye_0) \\ 2(e_xe_y-e_ze_0) & e_y^2+e_0^2-e_x^2-e_z^2 & 2(e_ye_z+e_xe_0) \\ 2(e_xe_z+e_ye_0) & 2(e_ye_z-e_xe_0) & e_z^2+e_0^2-e_x^2-e_y^2 \end{bmatrix} \tag{55}$$

Combining the diagonal components of Eq. (55) with Eq. (23) provides a four by four system of equations that is readily solved to relate the squares of the quaternion components to the diagonal components of the direction-cosine matrix,

$$\begin{Bmatrix} e_0^2 \\ e_x^2 \\ e_y^2 \\ e_z^2 \end{Bmatrix} = \frac{1}{4}\begin{Bmatrix} 1+C_{11}+C_{22}+C_{33} \\ 1+C_{11}-C_{22}-C_{33} \\ 1-C_{11}+C_{22}-C_{33} \\ 1-C_{11}-C_{22}+C_{33} \end{Bmatrix} \tag{56}$$

The off-diagonal components of Eq. (55) can be rearranged to give

$$\begin{Bmatrix} e_0e_x \\ e_0e_y \\ e_0e_z \\ e_xe_y \\ e_xe_z \\ e_ye_z \end{Bmatrix} = \frac{1}{4}\begin{Bmatrix} C_{23}-C_{32} \\ C_{31}-C_{13} \\ C_{12}-C_{21} \\ C_{12}+C_{21} \\ C_{31}+C_{13} \\ C_{23}+C_{32} \end{Bmatrix} \tag{57}$$

The components of the quaternion can be computed, without singularities, from the direction cosines by using Eqs. (56) and (57). This is done by first finding the quaternion component of greatest magnitude from Eq. (56),

$$e_{\max}^2 = \max(e_0^2, e_x^2, e_y^2, e_z^2) \tag{58}$$

Once the component of largest magnitude has been determined, the quaternion is computed from one of the following algorithms:

$$\begin{aligned} &\text{if}(e_0^2 = e_{\max}^2) \\ &\quad e_0 = \pm\sqrt{1+C_{11}+C_{22}+C_{33}}/2 \\ &\quad e_x = (C_{23}-C_{32})/4e_0 \\ &\quad e_y = (C_{31}-C_{13})/4e_0 \\ &\quad e_z = (C_{12}-C_{21})/4e_0 \end{aligned} \tag{59}$$

$$\begin{aligned} &\text{if}(e_x^2 = e_{\max}^2) \\ &\quad e_x = \pm\sqrt{1+C_{11}-C_{22}-C_{33}}/2 \\ &\quad e_0 = (C_{23}-C_{32})/4e_x \\ &\quad e_y = (C_{12}+C_{21})/4e_x \\ &\quad e_z = (C_{31}+C_{13})/4e_x \end{aligned} \tag{60}$$

$$\begin{aligned} &\text{if}(e_y^2 = e_{\max}^2) \\ &\quad e_y = \pm\sqrt{1-C_{11}+C_{22}-C_{33}}/2 \\ &\quad e_0 = (C_{31}-C_{13})/4e_y \\ &\quad e_x = (C_{12}+C_{21})/4e_y \\ &\quad e_z = (C_{23}+C_{32})/4e_y \end{aligned} \tag{61}$$

$$\begin{aligned} &\text{if}(e_z^2 = e_{\max}^2) \\ &\quad e_z = \pm\sqrt{1-C_{11}-C_{22}+C_{33}}/2 \\ &\quad e_0 = (C_{12}-C_{21})/4e_z \\ &\quad e_x = (C_{31}+C_{13})/4e_z \\ &\quad e_y = (C_{23}+C_{32})/4e_z \end{aligned} \tag{62}$$

Notice that, as was the case with Eq. (44), there are two possible quaternions that will satisfy Eq. (55). These

are the conventional right-hand Euler-Rodrigues quaternion and the left-hand quaternion that was previously mentioned.

Applying Rotational Constraints to the Quaternion Formulation

It is occasionally desirable to simulate aircraft motion with one or more of the rotational degrees of freedom constrained. This is normally accomplished using the Euler angle formulation. However, since the quaternion transformation is more than an order of magnitude faster than the Euler angle transformation, the ability to apply rotational constraints to the quaternion formulation may be of some interest.

When applying rotational constraints, the coordinate system in which the constraint will be applied must be specified. For example, a roll constraint is very different from a bank angle constraint. The roll can be constrained very simply, in either the Euler angle formulation or the quaternion formulation, by simply replacing the x_b-component of the angular momentum equation with the **roll constraint**

$$p = 0 \tag{63}$$

However, it is noted that constraining the roll does not constrain the bank angle. All three Euler angles can still take any possible value with the roll completely constrained.

To constrain the bank angle, using the quaternion formulation, Eq. (52) is employed. If the bank angle is to remain zero, Eq. (52) requires

$$e_0 e_x + e_y e_z = 0 \tag{64}$$

and

$$e_x \dot{e}_0 + e_0 \dot{e}_x + e_z \dot{e}_y + e_y \dot{e}_z = 0 \tag{65}$$

Applying Eq. (32) to Eq. (65) produces

$$\begin{aligned} & e_x(-pe_x - qe_y - re_z) \\ & + e_0(pe_0 + re_y - qe_z) \\ & + e_z(qe_0 - re_x + pe_z) \\ & + e_y(re_0 + qe_x - pe_y) = 0 \end{aligned} \tag{66}$$

This can be simplified to yield the **quaternion bank angle constraint**

$$(e_0^2 + e_z^2 - e_x^2 - e_y^2)p + 2(e_0 e_y - e_x e_z)r = 0 \tag{67}$$

Thus, to constrain the bank angle, the roll and the yaw must be coordinated according to Eq. (67).

If the roll is to be constrained as well as the bank angle, then Eq. (67) requires

$$2(e_0 e_y - e_x e_z)r = 0 \tag{68}$$

From Eq. (52), this is equivalent to

$$\sin(\theta)r = 0 \tag{69}$$

Here it is observed that, if both the roll and the bank angle are to be constrained, then either the yaw or the elevation angle must also be constrained. With any non-zero elevation angle, any amount of yaw will produce a bank angle. This is not a function of the quaternion formulation. It is a simple kinematic fact.

The relationship between the body-fixed angular rates and the Euler angles, which was demonstrated here by using the roll and bank angle, is true in general. None of the Euler angles can be constrained by constraining only one of the body-fixed angular rates. The bank angle can be changed using only pitch and yaw; the elevation angle can be changed using only roll and yaw; and the azimuth angle can be changed using only roll and pitch. *Any motion having only one of the rotational degrees of freedom constrained, in body-fixed coordinates, allows for all three degrees of freedom in orientation.*

Similarly, the other rotational degrees of freedom can be constrained using the **pitch constraint**

$$q = 0 \tag{70}$$

the **quaternion elevation angle constraint**

$$(e_0^2 + e_z^2 - e_x^2 - e_y^2)q - 2(e_0 e_x + e_y e_z)r = 0 \tag{71}$$

the **yaw constraint**

$$r = 0 \tag{72}$$

and the **quaternion azimuth angle constraint**

$$2(e_0 e_x - e_y e_z)q + (e_0^2 + e_y^2 - e_x^2 - e_z^2)r = 0 \tag{73}$$

Motion with only one rotational degree of freedom, in body-fixed coordinates, can also be simulated using the quaternion formulation. *Constraining any two rotational degrees of freedom in body-fixed coordinates will constrain two degrees of freedom in orientation.* For pure rolling motion,

$$q = r = 0 \tag{74}$$

Using these constraints in Eq. (32) gives

$$\begin{Bmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{Bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & -p & 0 & 0 \\ p & 0 & 0 & 0 \\ 0 & 0 & 0 & p \\ 0 & 0 & -p & 0 \end{bmatrix}\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} = \frac{p}{2}\begin{Bmatrix} -e_x \\ e_0 \\ e_z \\ -e_y \end{Bmatrix} \tag{75}$$

If the elevation angle and the azimuth angle are both initially zero, from Eq. (51), the initial condition for the quaternion is

$$\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix}_{t=0} = \begin{Bmatrix} \cos(\phi_0/2) \\ \sin(\phi_0/2) \\ 0 \\ 0 \end{Bmatrix} \tag{76}$$

Using Eq. (76) with Eq. (75), it is noted that both e_y and e_z remain zero during this motion. Thus, **pure rolling motion** can be simulated using the constraints

$$q = r = e_y = e_z = 0 \tag{77}$$

Similarly, **pure pitching motion** could be simulated by applying the quaternion constraints

$$p = r = e_x = e_z = 0 \tag{78}$$

and **pure yawing motion** results from the constraints

$$p = q = e_x = e_y = 0 \tag{79}$$

A Closed-form Quaternion Solution for Constant Rotation

It is possible to obtain a closed-form solution to the quaternion formulation for the case of constant rotation. While such a condition would frequently occur in spacecraft applications, it almost never occurs in aircraft applications. The angular rates experienced by a moving aircraft depend on the aerodynamic moments acting on the craft and are almost always changing with time. In this case, closed-form solutions to the quaternion formulation are very difficult or impossible to obtain, for all but the most trivial conditions.[156] Nevertheless, closed-form analytic solutions to the quaternion formulation for the case of constant angular rates do provide an excellent mechanism for verifying[157,158] the numerical algorithms used to integrate the quaternion formulation. For such verification, it is useful to have a general solution that allows for rotation about any or all of the body-fixed axes. The development of such a solution, from Eq. (32), is somewhat tedious but rather straightforward. The highlights of this development are presented here.

Consider a rigid body that is rotating at a constant angular velocity. The differential equations governing the change in the quaternion components with time are given by Eq. (32). This system can be written in differential operator notation as

$$\begin{bmatrix} 2D & -p & -q & -r \\ p & 2D & r & -q \\ q & -r & 2D & p \\ r & q & -p & 2D \end{bmatrix}\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{80}$$

The eigenvalues associated with the homogeneous system of differential equations given by Eq. (80) are obtained from

$$\begin{vmatrix} 2\lambda & -p & -q & -r \\ p & 2\lambda & r & -q \\ q & -r & 2\lambda & p \\ r & q & -p & 2\lambda \end{vmatrix} = 0 \tag{81}$$

which can be expanded and simplified to yield

$$(4\lambda^2 + p^2 + q^2 + r^2)^2 = 0 \tag{82}$$

Thus, the four eigenvalues associated with Eq. (80) consist of two identical complex pairs, both $\pm i\omega/2$, where $i = (-1)^{1/2}$ and $\omega^2 = p^2 + q^2 + r^2$. The possible solutions are then

$$\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} = \begin{Bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{Bmatrix} e^{i\omega t/2} + \begin{Bmatrix} C_5 \\ C_6 \\ C_7 \\ C_8 \end{Bmatrix} e^{-i\omega t/2} + \begin{Bmatrix} C_9 \\ C_{10} \\ C_{11} \\ C_{12} \end{Bmatrix} t e^{i\omega t/2} + \begin{Bmatrix} C_{13} \\ C_{14} \\ C_{15} \\ C_{16} \end{Bmatrix} t e^{-i\omega t/2} \tag{83}$$

Using the first of these solutions in Eq. (80) produces

$$\begin{bmatrix} i\omega & -p & -q & -r \\ p & i\omega & r & -q \\ q & -r & i\omega & p \\ r & q & -p & i\omega \end{bmatrix}\begin{Bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \tag{84}$$

Solving Eq. (84) by direct elimination gives only two independent relations among the 16 unknown constants

$$\begin{bmatrix} iu_x & iu_y & iu_z \\ \begin{pmatrix} R_x - \\ i(u_y - u_z) \end{pmatrix} & \begin{pmatrix} R_y - \\ i(u_z - u_x) \end{pmatrix} & \begin{pmatrix} R_z - \\ i(u_x - u_y) \end{pmatrix} \end{bmatrix} \begin{Bmatrix} C_2 \\ C_3 \\ C_4 \end{Bmatrix} = \begin{Bmatrix} C_1 \\ 0 \end{Bmatrix} \quad (85)$$

where $u_x=p/\omega$, $u_y=q/\omega$, $u_z=r/\omega$, and $R_i=u_i(u_x+u_y+u_z)-1$. Similarly, using the second solution from Eq. (83) in Eq. (80) yields two more independent relations

$$\begin{bmatrix} -iu_x & -iu_y & -iu_z \\ \begin{pmatrix} R_x + \\ i(u_y - u_z) \end{pmatrix} & \begin{pmatrix} R_x + \\ i(u_z - u_x) \end{pmatrix} & \begin{pmatrix} R_x + \\ i(u_x - u_y) \end{pmatrix} \end{bmatrix} \begin{Bmatrix} C_6 \\ C_7 \\ C_8 \end{Bmatrix} = \begin{Bmatrix} C_5 \\ 0 \end{Bmatrix} \quad (86)$$

Using the last two solutions from Eq. (83) in Eq. (80) yields only the trivial results that C_9 through C_{16} are all zero. The initial conditions provide four additional independent relations

$$\begin{Bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{Bmatrix} + \begin{Bmatrix} C_5 \\ C_6 \\ C_7 \\ C_8 \end{Bmatrix} = \begin{Bmatrix} e_0(0) \\ e_x(0) \\ e_y(0) \\ e_z(0) \end{Bmatrix} \quad (87)$$

By combining Eqs. (85) through (87), the eight remaining constants, C_1 through C_8, can be obtained. This gives the complete solution

$$\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} = \frac{1}{2} \begin{Bmatrix} e_0(0) + i[u_x e_x(0) + u_y e_y(0) + u_z e_z(0)] \\ e_x(0) - i[u_x e_0(0) + u_z e_y(0) - u_y e_z(0)] \\ e_y(0) - i[u_y e_0(0) - u_z e_x(0) + u_x e_z(0)] \\ e_z(0) - i[u_z e_0(0) + u_y e_x(0) - u_x e_y(0)] \end{Bmatrix} e^{i\omega t/2} + \frac{1}{2} \begin{Bmatrix} e_0(0) - i[u_x e_x(0) + u_y e_y(0) + u_z e_z(0)] \\ e_x(0) + i[u_x e_0(0) + u_z e_y(0) - u_y e_z(0)] \\ e_y(0) + i[u_y e_0(0) - u_z e_x(0) + u_x e_z(0)] \\ e_z(0) + i[u_z e_0(0) + u_y e_x(0) - u_x e_y(0)] \end{Bmatrix} e^{-i\omega t/2} \quad (88)$$

For the special initial condition where all three Euler angles are zero,

$$\begin{Bmatrix} e_0(0) \\ e_x(0) \\ e_y(0) \\ e_z(0) \end{Bmatrix} = \begin{Bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \quad (89)$$

the solution given by Eq. (88) reduces to the rather obvious result

$$\begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} = \begin{Bmatrix} (e^{i\omega t/2} + e^{-i\omega t/2})/2 \\ iu_x(e^{-i\omega t/2} - e^{i\omega t/2})/2 \\ iu_y(e^{-i\omega t/2} - e^{i\omega t/2})/2 \\ iu_z(e^{-i\omega t/2} - e^{i\omega t/2})/2 \end{Bmatrix} = \begin{Bmatrix} \cos(\omega t/2) \\ u_x \sin(\omega t/2) \\ u_y \sin(\omega t/2) \\ u_z \sin(\omega t/2) \end{Bmatrix} \quad (90)$$

Equations (88) and (90) provide an excellent mechanism for testing numerical algorithms used to integrate the quaternion formulation.

Numerical Integration of the Quaternion Formulation

When integrating Eq. (32) numerically, the zeros on the diagonal can cause some problems. Integration of this system can result in very large numerical errors if a first-order method is used. For this reason, a fourth-order method, such as fourth-order Runge-Kutta or fourth-order Adams-Bashforth-Moulton[159,160] should always be used to integrate the quaternion formulation. Since most modern numerical codes use such methods, this is not at all restrictive for simulations run on modern digital computers.

Historically, aircraft simulations using the quaternion formulation were first commonly run using analog computers. Since these analog simulations were inherently first-order, the raw quaternion formulation did not work well. When such simulations were run, the magnitude of the quaternion would grow quite rapidly with time. Since the formulation requires the magnitude of the quaternion to remain unity in order to maintain orthogonality in the attitude transformation, these large orthogonality errors would render the simulations almost useless. To remedy this problem, techniques were developed to reduce this orthogonality error. One such method, developed by Robinson[36,37] and popularized by Mitchell and Rogers[38], is based on the method first suggested by Corbett and Wright[39] for maintaining orthogonality in the direction-cosine matrix. With this method, the kinematic equations in

Eq. (32) were modified to include error reduction terms on the diagonal,

$$\begin{Bmatrix} \dot{e}_0 \\ \dot{e}_x \\ \dot{e}_y \\ \dot{e}_z \end{Bmatrix} = \frac{1}{2} \begin{bmatrix} k\varepsilon & -p & -q & -r \\ p & k\varepsilon & r & -q \\ q & -r & k\varepsilon & p \\ r & q & -p & k\varepsilon \end{bmatrix} \begin{Bmatrix} e_0 \\ e_x \\ e_y \\ e_z \end{Bmatrix} \tag{91}$$

where ε is the orthogonality error defined as,

$$\varepsilon = 1 - (e_0^2 + e_x^2 + e_y^2 + e_z^2) \tag{92}$$

and k is a gain coefficient, which Mitchell and Rogers[38] said should be "set to a very high value." This error reduction scheme, which shall be called *Corbett-Wright orthogonality control*, worked quite well when used with the analog computers of that era.

While analog computers are no longer widely available, it is possible to demonstrate approximately how Corbett-Wright orthogonality control worked with an analog computer by integrating Eq. (91) using a first-order Euler integration scheme. Results of such an integration are shown in Fig. 2. For this figure the total error was computed as the difference between the numerical result and the exact solution from Eq. (88). Notice that, as the value of the Corbett-Wright gain coefficient approaches zero, which is the case of no orthogonality control, the total error becomes very large. For gain coefficients larger than about 0.5ω the error is reduced to a more or less acceptable level. With the exception of the increase in error on the right, the results shown in Fig. 2 are very similar to the results reported by Mitchell and Rogers[38] for a typical analog computer. Unlike the first-order digital integration,

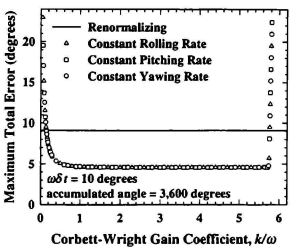


Fig. 2. The effect of changing the gain coefficient on the accuracy of a first-order Euler algorithm using Corbett-Wright orthogonality control.

analog simulations were well behaved for very large values of k. The abrupt increase in error that is seen in Fig. 2 for k/ω greater than about 5.7 is a result of the well known Euler instability that is associated with any first-order numerical integration. For Eq. (91), Fang and Zimmerman[161] have shown that this first-order instability occurs whenever the product of the gain coefficient and the time step is greater than unity,

$$k\,\delta t > 1$$

When digital computers first became fast enough to replace analog computers for running such simulations, Corbett-Wright orthogonality control was often used in the digital solution algorithms. Furthermore, this orthogonality control is still widely used for aircraft flight simulation today.[33,40] It is, however, not necessary. When a modern digital computer is used to integrate Eq. (32) with any of the prevalent fourth-order numerical methods, the orthogonality error is extremely small. Using the Corbett-Wright orthogonality control scheme with these modern numerical algorithms increases the computation time but does little to improve the accuracy of the simulation.

Even though the orthogonality error for modern numerical algorithms is very small, it can accumulate. Thus, if a simulation is to be run for long periods of time using large time steps, the quaternion should be renormalized periodically. The quaternion may be renormalized by dividing each component by the magnitude,[153,162,163]

$$\mathbf{e} = \frac{\mathbf{e}}{|\mathbf{e}|} = \frac{\mathbf{e}}{\sqrt{e_0^2 + e_x^2 + e_y^2 + e_z^2}} \tag{93}$$

As can be seen in Fig. 2, renormalization is not as accurate as Corbett-Wright orthogonality control for first-order integration. However, a different result is obtained for higher-order integration.[164] If the time steps used with fourth-order Runge-Kutta are of a size to produce about 10 degrees of revolution per time step, it requires about 3,000,000 iterations to degrade the magnitude of the quaternion by one percent. Since a very long renormalization period can be used, periodic renormalization of the quaternion is much faster than using the Corbett-Wright orthogonality control scheme. However, if computation time is not a concern, Corbett-Wright orthogonality control will eliminate the growth of the orthogonality error for simulations using large time steps. If the time steps are small enough to produce about five degrees of rotation or less of per time step, no orthogonality control whatsoever is required with fourth-order Runge-Kutta integration.

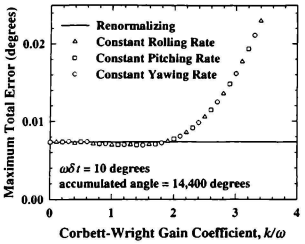


Fig. 3. The effect of changing the gain coefficient on the accuracy of a fourth-order Runge-Kutta algorithm using Corbett-Wright orthogonality control.

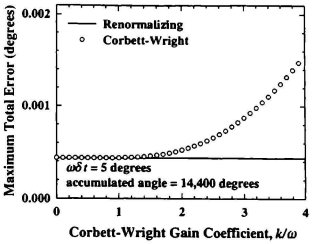


Fig. 5. The effect of reducing the step size on the accuracy of a fourth-order Runge-Kutta algorithm using Corbett-Wright orthogonality control.

If the Corbett-Wright orthogonality control scheme is to be used for large time step simulations, it should be remembered that this only eliminates the growth of the orthogonality error. It does not eliminate the drift error, which for modern algorithms can be much larger than the orthogonality error. In fact, when a large gain coefficient is used with fourth-order integration and Corbett-Wright orthogonality control, the drift error is increased by more than the orthogonality error is reduced, which actually increases the net error for the simulation. This is shown in Fig. 3. In contrast with the older analog simulations that used a very large gain coefficient, when Corbett-Wright orthogonality control is used with a fourth-order digital integration, a gain coefficient larger than about 2ω should not be used.

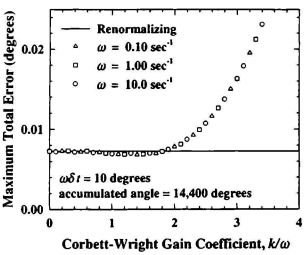


Fig. 4. The effect of changing the angular rate on the accuracy of a fourth-order Runge-Kutta algorithm using Corbett-Wright orthogonality control.

As can be seen in Fig. 4, this is truly independent of the angular rate. Furthermore, as is seen in Fig. 5, decreasing the time step does not eliminate this problem. For such fourth-order integration, a gain coefficient equal to the magnitude of the angular velocity will effectively eliminate the growth of the orthogonality error associated with large time steps without adversely affecting the drift error. However, for aircraft flight simulations, the angular rates are not typically known a priori. Furthermore, the angular rate is typically changing with time and a very low angular rate requires a very low gain coefficient for accurate simulation. Thus, to avoid increasing the total error, a variable gain coefficient equal to the angular rate should be used,

$$k = \omega = \sqrt{p^2 + q^2 + r^2} \tag{94}$$

The implementation of Eq. (94) adds substantially to the computational burden of Corbett-Wright orthogonality control. Thus, for greatest computational efficiency, it is much preferred to use periodic renormalization of the quaternion over any use of the Corbett-Wright orthogonality control scheme.

In 1969 Fang and Zimmerman[161] proposed a pseudo fourth-order method for integrating Eq. (91). With conventional fourth-order Runge-Kutta the time rate of change of the quaternion is computed from Eq. (91) four times for each time step. With the Fang and Zimmerman algorithm this same fourth-order procedure is used to integrate Eq. (91), except that ε is held constant through all four computations in each time step. The orthogonality error is updated only once at the beginning of each full time step.

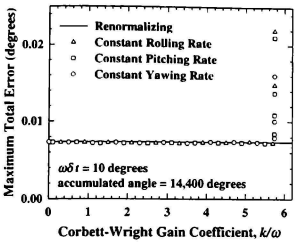


Fig. 6. The effect of changing the gain coefficient on the accuracy of the Fang and Zimmerman algorithm using Corbett-Wright orthogonality control.

The Fang and Zimmerman algorithm exhibits features of both first-order and fourth-order numerical integration. Although both the angular velocity and the quaternion on the right-hand side of Eq. (91) are estimated to fourth-order accuracy, ε is only estimated to first-order accuracy. Results of integrating Eq. (91) using the Fang and Zimmerman algorithm are shown in Fig. 6. Notice that the accuracy of the method, for small values of the gain coefficient, is comparable to that for full fourth-order integration. However, the method also exhibits an abrupt first-order instability when the gain coefficient is greater than one divided by the time step, $k > 1/\delta t$.

One striking observation that should be noted from Figs. 3 through 6 is that Corbett-Wright orthogonality control offers no significant reduction in simulation error for modern numerical integration algorithms. On the other hand, if the gain coefficient is not judicially chosen, Corbett-Wright orthogonality control can significantly increase the simulation error. Periodic renormalization of the quaternion does not present this problem and is much more computationally efficient.

The computation time required for periodic renormalization of the quaternion can be reduced even further by recognizing that the orthogonality error is always very small for modern fourth-order integration algorithms. Equation (93) can be written as

$$\mathbf{e} = \frac{\mathbf{e}}{\sqrt{e_0^2 + e_x^2 + e_y^2 + e_z^2}} = \frac{\mathbf{e}}{\sqrt{1-\varepsilon}} \qquad (95)$$

where ε is the error in the square of the quaternion magnitude, as given by Eq. (92). Since ε is very small, Eq. (95) can be closely approximated as[16]

$$\mathbf{e} = \mathbf{e}\left(1 + \tfrac{1}{2}\varepsilon\right) \qquad (96)$$

or after applying Eq. (92)

$$\mathbf{e} = \mathbf{e}\left[1.5 - 0.5\left(e_0^2 + e_x^2 + e_y^2 + e_z^2\right)\right] \qquad (97)$$

Periodic application of Eq. (97) is sufficient to eliminate the growth of the orthogonality error for all time.

Renormalizing the quaternion with Eq. (97) requires 9 multiplications and 4 additions. Using Eq. (93), on the other hand, requires 8 multiplications, 3 additions, 1 square root, and 1 division. Typical modern hardware requires 1 instruction cycle for addition, 1 instruction cycle for multiplication, something on the order of 4 instruction cycles for division, and anywhere from 4 to 30 instruction cycles for the square root operation, depending on the processor and the compiler. Thus, renormalizing with Eq. (93) requires anywhere from about 50 to 250 percent greater computation time than using Eq. (97), on typical modern hardware. Furthermore, even if future developments reduce the time required for division and square root to 1 instruction cycle each, Eq. (97) will be, at worst, computationally equivalent to Eq. (93).

While periodic renormalization of the quaternion using Eq. (97) provides a computationally efficient means for controlling the orthogonality error, it does not control the drift error. In fact, for higher-order algorithms, it does nothing whatsoever to reduce the total error. In Figs. 3 through 6, the points that correspond to a Corbett-Wright gain coefficient of zero were obtained using no orthogonality control at all. In each case the total error obtained with no orthogonality control was the same as that obtained when the quaternion was renormalized after each time step.

Conclusions

Among some fraction of the aircraft community, the quaternion formulation has gained the somewhat undeserved reputation of being hard to understand. This reputation has more to do with the scattered nature of the literature on this topic, particularly in aircraft journals and textbooks, than it does with the raw complexity of the quaternion formulation itself. Information from over 160 references, pertinent to attitude kinematics, was assembled in an attempt to provide a clearer understanding of the quaternion formulation. To this end, a systematic development of the kinematic transformation equations in terms of Euler angles, direction cosines, the Euler axis rotation parameters, and the Euler-Rodrigues quaternion was presented. The kinematic equations, including the

gravitational vector and the effects of wind were presented for these four attitude representations. The singularities in the both the Euler angle formulation and the Euler axis formulation were discussed. Since the physical interpretation of the Euler-Rodrigues quaternion is much less intuitive than that associated with Euler angles, relationships between the quaternion and the other attitude descriptors were presented.

The singularity associated with the Euler angle formulation has often lead the aircraft community to implement either the direction cosine formulation or the quaternion formulation for the description of attitude. In addition to eliminating this singularity, it is argued that an important advantage of the quaternion formulation over either the Euler angle formulation or the direction cosine formulation is increased computational speed. Numerical integration of the nine component direction-cosine formulation is excessively time consuming when compared with the four component quaternion formulation. Through the use of quaternion algebra, the computational advantage of the Euler-Rodrigues formulation can be further extended. Utilizing quaternion algebra, it was shown that the Euler angle transformation requires about 11 times as long to evaluate as the quaternion transformation. Consequently, even in cases where the aircraft is not an all-attitude vehicle and the gimbal lock singularity is not encountered, the quaternion formulation provides important computational savings when compared with either the Euler angle formulation or the direction-cosine formulation.

In order to take full advantage of the speed and accuracy of the Euler-Rodrigues quaternion formulation, it is necessary to give some consideration to the numerical method used to integrate the kinematic equations. Early aircraft flight simulations were run on analog computers that were inherently first-order. Numerical integration of the quaternion rate equations with a first-order method results in very large errors. Early analog programmers knew of this behavior and they employed an error reduction scheme, which in this paper was called Corbett-Wright orthogonality control. This orthogonality control is still found in some aircraft simulation codes in use today. Integration with any of today's prevalent fourth-order numerical methods produces very little orthogonality error. Using the Corbett-Wright orthogonality control scheme with modern numerical algorithms increases the computational time but does little to improve the accuracy of the simulation.

Even though the orthogonality error for modern numerical algorithms is very small, it can accumulate. Periodic renormalization of the quaternion eliminates this error. The Corbett-Wright orthogonality control scheme will also eliminate the growth of the orthogonality error, but it does not eliminate the drift error. In some cases, for certain values of the gain coefficient, the drift error is increased by more than the orthogonality error is reduced which actually increases the total error for the simulation. Periodic renormalization of the quaternion provides a computationally efficient means for controlling orthogonality error when compared with Corbett-Wright, but it does not control drift error either. Reduction in size of the time step and/or increasing the order of the integration are the only effective ways to reduce total error.

Acknowledgement

The authors appreciate the editorial and bibliographic efforts of Ms Joan P. Smith.

References

[1]Whitaker, E. T., *Analytical Dynamics*, Dover, New York, 1944, pp. 1-14.

[2]Goldstein, H., *Classical Mechanics*, 2nd ed., Addison-Wesley, Reading, MA, 1981, pp. 148-158.

[3]Nakano, A., "Rigid-body-based Multiple Time Scale Molecular Dynamics Simulation of Nanophase Materials," *International Journal of High Performance Computing Applications*, Vol. 13, No. 2, 1999, pp. 154-162.

[4]Hover, F. S., "Simulation of Massless Tethers," *Ocean Engineering*, Vol. 24, No. 8, 1997, pp. 765-783.

[5]Smith, R., Frost, A., and Probert, P., "A Sensor System for the Navigation of an Underwater Vehicle," *The International Journal of Robotics Research*, Vol. 18, No. 7, 1999, pp. 697-710.

[6]Fjellstad, O. E. and Fossen, T. I., "Position and Attitude Tracking of AUV's: A Quaternion Feedback Approach," *IEEE Journal of Oceanic Engineering*, Vol. 19, No. 4, 1994, pp. 512-518.

[7]Chou, J. C. K., and Kamel, M., "Finding the Position & Orientation of a Sensor on a Robot Manipulator Using Quaternions," *International Journal of Robotics Research*, Vol. 10, No. 3, 1991, pp. 240-254.

[8]Wee, L. G., Walker, M. W., and McClamroch, N. H., "An Articulated-Body Model for a Free-Flying Robot and its Use for Adaptive Motion Control," *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 2, 1997, pp. 264-277.

[9]Dornaika, F. and Horaud, R., "Simultaneous Robot-World and Hand-Eye Calibration," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 4, 1998, pp. 617-622.

[10]Caccavale, F., Natale, C., Siciliano, B., and Villani, L., "Six-DOF Impedance Control Based on Angle/Axis Representations," *IEEE Transactions on Robotics and Automation*, Vol. 15, No. 2, 1999, pp. 289-300.

[11]Daniilidis, K., "Hand-eye Calibrations Using Dual Quaternions," *International Journal of Robotics Research*, Vol. 18, No. 3, 1999, pp. 286-98.

[12]Omelyan, I. P., "On the Numerical Integration of Motion for Rigid Polyatomics: the Modified Quaternion Approach," *Computers in Physics*, Vol. 12, No. 1, 1998, pp. 97-103.

[13]Kuipers, J. B. "SPASYN An Electromagnetic Relative Position and Orientation Tracking System," *IEEE Transactions on Instrumentation and Measurements*, Vol. 29, No. 4, 1980, pp. 462-471.

[14]Shoemake, K., "Animating Rotation with Quaternion Curves," *Computer Graphics*, Vol. 19, No. 3, 1985, pp. 245-254.

[15]Hanson, A. J. and Ma, H., "Quaternion Frame Approach to Streamline Visualization," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 1., No. 2, 1995, pp. 164-174.

[16]Hart, J. C., Francis, G. K., and Kauffman, L. H., "Visualizing Quaternion Rotation," *ACM Transactions on Graphics*, Vol. 13, No. 3, 1994, pp. 256-276.

[17]Chou, J. C. K., "Quaternion Kinematics and Dynamic Differential Equations, " *IEEE Transactions on Robotics & Automation*, Vol. 8, No. 1, 1992, pp. 53-64.

[18]Shuster, M. D., "A Survey of Attitude Representations," *The Journal of the Astronautical Sciences*, Vol. 14, No. 4, 1993, pp. 439-517.

[19]Perkins, C. D., and Hage, R. E., *Airplane Performance Stability & Control*, Wiley, New York, 1949, pp. 374-474.

[20]Duncan, W. J., *The Principles of Control and Stability of Aircraft*, Cambridge University Press, Cambridge, 1952, pp. 61-92.

[21]Babister, A. W., *Aircraft Stability and Control*, Oxford University Press, Oxford, 1961, pp. 1-700.

[22]Kolk, W. R., *Modern Flight Dynamics*, Prentice-Hall, Englewood Cliffs, NJ, 1961, pp. 22-40.

[23]Etkin, B., *Dynamics of Atmospheric Flight*, Wiley, New York, 1972, pp. 121-154.

[24]McRuer, D. T., Ashkenas, I. L., and Graham, D., *Aircraft Dynamics and Automatic Control*, Princeton University Press, Princeton, NJ, 1973, pp. 203-295.

[25]Babister, A. W., *Aircraft Dynamic Stability and Response*, Pergamon, Oxford, 1980, pp. 11-28.

[26]McCormick, B. W., *Aerodynamics, Aeronautics, and Flight Mechanics*, 2nd ed., Wiley, New York, 1995, pp. 538-582.

[27]Etkin, B., and Reid L. D., *Dynamics of Flight: Stability and Control*, 3rd ed., Wiley, New York, 1996, pp. 93-128.

[28]Cook, M. V., *Flight Dynamics Principles*, Wiley, New York, 1997, pp. 11-29.

[29]Nelson, R. C., *Flight Stability & Automatic Control*, 2nd ed., McGraw-Hill, New York, 1998, pp. 96-130.

[30]Roskam, J., *Airplane Flight Dynamics and Automatic Flight Control*, DAR Corporation, Lawrence, KS, 1998, pp. 3-34.

[31]Schmidt, L. V., *Introduction to Aircraft Flight Dynamics*, AIAA, Reston, VA, 1998, pp. 93-118.

[32]Abzug, M. J., *Computational Flight Dynamics*, AIAA, Reston, VA, 1998, pp. 5-7.

[33]Pamadi, B. N., *Performance, Stability, Dynamics, and Control of Airplanes*, AIAA, Reston, VA, 1998, pp. 344-347.

[34]Abzug, M. J. and Larrabee, E. E., *Airplane Stability and Control: A History of Technologies that Made Aviation Possible*, Cambridge University Press, Cambridge, 1997, pp. 251-259.

[35]Jackson, E. B., "Manual for a Workstation-based Generic Flight Simulation Program (LaRCsim) Version 1.4," NASA TM-110164, April 1995.

[36]Robinson, A. C., "On the Use of Quaternions in the Simulation of Rigid Body Motion," *Proceedings of Simulation Council Conference*, Washington, DC, Dec. 1958, pp. 1-18.

[37]Robinson, A. C., "On the Use of Quaternions in the Simulation of Rigid Body Motion," WADC TR 58-17, Wright Patterson AFB, OH, Dec. 1958.

[38]Mitchell, E. E. L. and Rogers, A. E., "Quaternion Parameters in the Simulation of a Spinning Rigid Body," *Simulation*, Vol. 4, No. 6, 1965, pp. 390-396.

[39]Corbett, J. P., and Wright, F. B., "Stabilization of Computer Circuits," WADC TR 57-425, E. Hochfeld, editor, Wright Patterson AFB, OH, Nov. 1957.

[40]Rolfe, J. M. and Staples, K. J., *Flight Simulation*, Cambridge University Press, Cambridge, 1988, pp. 380-440.

[41]Kaplan, M. H., *Modern Spacecraft Dynamics and Control*, Wiley, New York, 1976, pp. 1-415.

[42]Messiah, A., *Quantum Mechanics*, North-Holland Publishing Company, Amsterdam, Vol. 1, 1961, pp. 1-504 and Vol. 2, 1962 pp. 505-1136.

[43]Tandon, G. K., "Coordinate Transformations," in *Spacecraft Attitude Determination and Control*, J. R. Wertz, editor, Kluwer Academic Publishers, Dordrecht, 1978, pp. 760-766.

[44]Pio, R. L., "Symbolic Representation of Coordinate Transformation," *IEEE Transactions on Aerospace and Navigational Electronics*, Vol. 11, June 1964, 128-134.

[45]Pio, R. L., "Euler Angle Transformations," *IEEE Transactions on Automatic Control*, Vol. 11, No. 4, 1966, pp. 707-715.

[46]Bar-Itzhack, I. Y., "The Program: An Essay Tool for Angular Position and Rate Computations," *Journal of the Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 519-529.

[47]Mayer, A., "Rotations and Their Algebra," *SIAM Review*, Vol. 2, No. 2, 1960, pp. 77-122.

[48]Hughes, P. C., *Spacecraft Attitude Dynamics*, Wiley, New York, 1986, pp. 1-564.

[49]Kane, T. R., Likins, P. W., and Levinson, D. A., *Spacecraft Dynamics*, McGraw Hill, New York, 1983, pp. 1-436.

[50]Junkins, J. L., and Shuster, M. D., "The Geometry of the Euler Angles," *Journal of the Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 531-543.

[51]Markley, F. L., "New Dynamic Variables for Momentum-Bias Spacecraft," *Journal of the Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 557-567.

[52]Broucke, R. A., "On the Use of Poincare Surfaces of Section in Rigid Body Motion," *Journal of Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 593-601.

[53]Kolve, D. I., "Describing an Attitude," *Advances in the Astronautical Sciences, Proceedings of the 16th AAS Rocky Mountain Guidance and Control Conference*, Feb. 1992, Vol. 81, 1993, Keystone, CO, Published by Univelt Inc., San Diego CA, pp. 289-303.

[54]Markley, F. L., "Attitude Dynamics," in *Spacecraft Attitude Determination and Control*, J. R. Wertz, editor, Kluwer Academic Publishers, Dordrecht, 1978, pp. 510-521.

[55]Sabroff, A., Farrenkopf, R., Frew, A., and Gran, M., AFFDL-TR-65-115, TRW, Redondo Beach, CA, 1965.

[56]Sidi, M. J., *Spacecraft Dynamics & Control*, Cambridge University Press, Cambridge, 1997, pp.318-327.

[57]Satake, I., *Linear Algebra*, Marcel Dekker, Inc., New York, 1975, pp. 198-208.

[58]Bryan, T. R., Lewis, D. G. and Perry, D. D., "Software User's Manual for TRAP 3.1," TAG 92-03, Battelle, Columbus, OH, March, 1993.

[59]Euler, L., "Formulae Generales pro Translatione Quacunque Corporum Rigidorum," *Novi Commentari Academiae Scientiarum Imperialis Petropolitanae*, Vol. 20, 1775, pp. 189-207.

[60]Euler L., "Nova Methodus Motum Corporum Rigidorum Determinandi," *Novi Commentari Academiae Scientiarum Imperialis Petropolitanae*, Vol. 20, 1775, pp. 208-238.

[61]Beatty, M. F., "Vector Analysis of Finite Rigid Rotations," *Journal of Applied Mechanics*, Vol. 44, No. 3, 1977, pp. 501-502.

[62]Kozik, T. J., "Finite Rotations and Associated Displacement Vectors," *Journal of Applied Mechanics*, Vol. 43, No. 4, 1976, pp. 698-699.

[63]Mathews, J., "Coordinate-Free Rotation Formalism," *American Journal of Physics*, Vol. 44, No. 12, 1976, pp. 1210-1210.

[64]Pearlman, N., "Vector Representation of Rigid-Body Rotations," *American Journal of Physics*, Vol. 35, No. 12, 1967, pp. 1164-1167.

[65]Beatty, M. F., "Kinematics of Finite, Rigid-Body Displacements," *American Journal of Physics*, Vol. 35, No. 10, 1967, pp. 949-954.

[66]Beatty, M. F., "Vector Representation of Rigid Body Rotation," *American Journal of Physics*, Vol. 31, No. 2, 1963, pp. 134-135.

[67]Grubin, C., "Vector Representation of Rigid-Body Rotation," *American Journal of Physics*, Vol. 30, No. 6, 1962, p. 416-417.

[68]Euler, L., "Problema Algebraicum ob Affectiones Prorsus Singulares Memorabile," *Novi Commentari Academiae Scientiarum Imperialis Petropolitanae*, Vol. 15, 1770, pp. 75-106.

[69]Rodrigues, O., "Des Lois Geometriques qui Regissent les Deplacements d'un Systeme Solide dans L'espace, et de la Variation des Coordonnees Provenant de Ses Deplacements Consideeres Independamment des Causes qui Peuvent les Produire," *Journal des Mathematiques Pures et Appliquees*, Vol. 5, 1840, pp. 380-440.

[70]Cayley, A., "On Certain Results Relating to Quaternions," *Philosophical Magazine*, Vol. 26, No. 1, 1845, pp. 141-145.

[71]Hamilton, W. R., "On Quaternions: Or a New System of Imaginaries in Algebra," *Philosophical Magazine*, Vol. 25, No. 3, 1844, pp. 489-495.

[72]Hamilton, W. R., *Lectures on Quaternions*, Hodges and Smith, Dublin, 1853, pp. 1-736.

[73]Hamilton, W. R., *Elements of Quaternions*, Longmans, Green & Co., London, 1866.

[74]Graves, R. P. *Life of Sir William Rowan Hamilton*, Three Volumes, Hodges, Figgis and Co, Dublin, 1882-1889.

[75]Hankins, T. L., *Sir William Rowan Hamilton*, Johns Hopkins University Press, Baltimore, 1980, pp. 1-474.

[76]O'Donnell, S., *William Rowan Hamilton, Portrait of a Prodigy*, Boole, Dublin Ireland, 1983, pp. 1-224.

[77]Altmann, S. L., *Rotations, Quaternions, and Double Groups*, Oxford University Press, Oxford, 1986, pp. 1-317.

[78]Altmann, S. L., "Hamilton, Rodrigues, and the Quaternion Scandal," *Mathematics Magazine*, Vol. 62, No. 5, 1989, pp. 291-307.

[79]Cheng, H., and Gupta, K. C., "An Historical Note on Finite Rotations," *ASME Journal of Applied Mathematics*, Vol. 56, No. 2, 1989, pp. 139-145.

[80]Euler, L., "Du Mouvement de Rotation des Corps Solides Autour d'un Axe Variable," *Memoires de l'academiei des sciences de Berlin*, Vol. 14, 1758, pp. 154-193.

[81]Euler, L., "DeMotu Corporum Circa Punctum Fixum Mobilium," *Commentatio 825 indicis Enestroemiani, Opera Posthuma*, Vol. 2, 1862, pp. 43-62.

[82]Roberson, R. E., "Kinematic Equations for Bodies Whose Rotation is Described by the Euler-Rodrigues Parameters," *AIAA Journal*, Vol. 6, No. 5, 1968, pp. 916-917.

[83]Jacobi, D. G. J., "Bemerkungen zu einer Abhandlung Euler's uber die orthogonale Substitution," *C. G. J. Jacobi's Gesammelte Werke*, Vol. III, Chelsea Publishing Company, New York, 1969, pp. 601-609.

[84]Grassman, H. G., *Die lineale Ausdehnungslehre, eom meier Zweig de Mathematik dargestellt und durch Anwendungen auf die ubrige Zweige de Mathematik, wie auch auf die Statik, Mechanik, die Lehre von Magnetismus und due Krystallonomie erlautert*, 1844, pp. 1-301.

[85]Grassman, H. G., *Die Ausdehnungslehre: vollstandig und in strenger Form bearbeitel*,1862,

[86]Gibbs, J. W., *Vector Analysis*, E. B. Wilson, editor, Scribners, New York, 1901.

[87]Heaviside, O., *Electromagnetic Theory*, Vol. 1, Dover, New York, 1950.

[88]Boyer, C. B., and Merzbach, U.C., *A History of Mathematics*, 2nd ed., Wiley, New York, 1989, pp. 656-762.

[89]Gray, J. J., "Olinde Rodrigues' Paper of 1840 on Transformation Groups," *Archive for the History of Exact Sciences*, Vol. 21, No. 4, 1980, pp. 375-385.

[90]Cartan, E., *Lecons Sur la Theorie des Spineurs. I. Les Spineurs de l'espace a Trois Dimensions*, Hermann, Paris, 1938, pp. 57-63.

[91]Temple, G., *Cartesian Tensors. An Introduction*, Methuen, London, 1960, pp. 68-92.

[92]Booth, A. J., *Saint-Simon and Saint-Simonism, A Chapter in the History of Socialism in France*, Longmans, Green, Reader and Dyer, London, 1871, pp. 1-262.

[93]Wilson, E., *To the Finland Station. A Study in the Writing and Acting of History*, Secker and Warburg, London, 1941, pp. 100-509.

[94]Kline, M., *Mathematical Thoughts from Ancient to Modern Times*, Oxford University Press, Oxford, 1972, pp. 1-1238.

[95]Van Der Waerden, B. L., *A History of Algebra*, Springer-Verlag, Heidelberg, 1985, pp. 1-271.

[96]Crowe, M. J., *A History of Vector Analysis*, Dover, New York, 1967, pp. 1-270.

[97]McDuffee, C. C., *The Theory of Matrices*, Chelsea Publishing Company, New York, 1946, pp. 1-110.

[98]Shuster, M. D., and Natanson, G. A., "Quaternion Computation from a Geometric Point of View," *Journal of the Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 545-556.

[99]Vathsal, S., "Derivation of the Relative Quaternion Differential Equations," *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 5, 1991, pp. 1061-1064.

[100]Stuelpnagel, J., "On the Parameterization of the Three-Dimensional Rotation Group," *SIAM Review*, Vol. 6, No. 4, 1964, pp. 422-430.

[101]Klumpp, A. R., "Singularity-free Extraction of a Quaternion from a Direction-Cosine Matrix," *Journal of Spacecraft and Rockets*, Vol. 13, No. 12, 1976, pp. 754-755.

[102]Spurrier, R. A., "Comment on 'Singularity-free Extraction of a Quaternion from a Direction-Cosine Matrix," *Journal of Spacecraft and Rockets*, Vol. 15, No. 4, 1978, pp. 255-256.

[103]Shepperd, S. W., "Quaternion from Rotation Matrix," *Journal of Guidance and Control*, Vol. 1, No. 3, 1978, pp. 223-224.

[104]Jiang, Y. F., and Lin, Y. P., "Error Analysis of Quaternion Transformation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 27, No. 4, 1991, pp. 634-638.

[105]Aramanovitch, L., "Spacecraft Orientation Based on Space Object Observations by Means of Quaternion Algebra," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 4, 1995, pp. 859-866.

[106]Kuipers, J. B., *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*, Princeton University Press, Princeton, NJ, 1999, pp. 155-168.

[107]Mayo, R. A. "Relative Quaternion State Transition Relation," *Journal of Guidance and Control*, Vol. 2, No. 1, 1979, pp. 44-48.

[108]Stevens, B. L. and Lewis, F. L, *Aircraft Control & Simulation*, Wiley, New York, 1992, pp. 39-44.

[109]McFarland, R. E., "A Standard Kinematic Model for Flight Simulation at NASA-Ames," NASA CR-2497, Jan. 1975.

[110]Jenkins, P. N., "Missile Dynamics Equations for Guidance and Control Modeling and Analysis," RG-84-17, Redstone Arsenal, AL, April 1984.

[111]Wilcox, J. C. "A New Algorithm for Strapdown Inertial Navigators," *IEEE Transactions on Aerospace & Electronic Systems*, Vol. 3, No. 5, 1967, pp. 796-802.

[112]Mckern, R. A., "A Study of Transformation Algorithms for Use in a Digital Computer," Master Thesis T-493, Massachusetts Institute of Technology, Cambridge, 1968.

[113]Bortz, J. E., "A New Mathematical Formulation for Strapdown Inertial Navigation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 7, No. 1, 1971, pp. 61-66.

[114]Mortensen, R. E., "Strapdown Guidance Error Analysis," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 10, No. 5, 1974, pp. 451-457.

[115]Friedland, B., "Analysis of Strapdown Navigation Using Quaternions," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 14, No. 5, 1989, pp. 764-768.

[116]Lovren, N., and Pieper, J. K., "Error Analysis of Direction Cosines and Quaternion Parameter Techniques for Aircraft Attitude Determination," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 34, No. 3, 1998, pp. 983-989.

[117]Lefferts, E. J., Markley, F. L., and Shuster, M. D., "Kalman Filtering For Spacecraft Attitude Estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 5, No. 5, 1982, pp. 417-429.

[118]Bar-Itzhack, I. Y., and Oshman, Y., "Attitude Determination from Vector Observations: Quaternion Estimation," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 21, No. 1, 1985, pp. 128-136.

[119]Shuster, M. D., "Quaternion on the Kalman Filter," *Advances in the Astronautical Sciences*, Vol. 85, No. 1, 1993, pp. 16-19.

[120]Creamer, G., DeLaHunt, P., and Gates, S., "Attitude Determination and Control of Clementine During Lunar Mapping," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 3, 1996, pp. 505-511.

[121]Ickes, B. F., "A New Method for Performing Digital Control System Attitude Computations Using Quaternions," *AIAA Journal*, Vol. 8, No. 1, 1970, pp. 13-17.

[122]Grubin, C., "Attitude Determination for a Strapdown Inertial System Using the Euler-Axis/Angle and Quaternion Parameters," AIAA Paper 73-900, 1973.

[123]Morton, H. S., Jr., Junkins, J. L., and Blanton, J. N., "Analytical Solutions for Euler Parameters," *Celestial Mechanics*, Vol. 10, 1974, pp. 287-301.

[124]Morton, H. S. Jr., "Hamiltonian and Lagrangian Formulations of Rigid Body Rotational Dynamics Based on the Euler Parameters," *Journal of Astronautical Sciences*, Vol. 41, No. 4, 1993, pp. 569-591.

[125]Vadali, S. R., and Junkins, J. L., "Optimal Open-Loop and Stable Feedback Control of Rigid Spacecraft Attitude Maneuvers," *Journal of the Astronautical Sciences*, Vol. 32, No. 1, 1984, pp. 105-122.

[126]Vadali, S. R., Kraige, L. G., and Junkins, J. L., "New Results on the Optimal Spacecraft Attitude Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 7, No. 3, 1984, pp. 378-380.

[127]Vathsal, S., "Optimal Control of Quaternion Propagation Errors in Spacecraft Navigation," *Journal of Guidance, Control, and Dynamics*, Vol. 9, No. 3, 1986, pp. 382-385.

[128]Wie, B., Weiss, H., and Arapostathis, A., "Quaternion Feedback Regulator for Spacecraft Eigenaxis Rotations," *Journal of Guidance, Control, and Dynamics*, Vol. 12, No. 3, 1989, pp. 375-380.

[129]Chatterji, G. B., and Tahk, M., "Quaternion Formulation for Boost Phase Attitude Estimation, Guidance and Control of Exoatmospheric Interceptors," *Proceedings of the 1989 American Control Conference*, IEEE, Pittsburgh, PA, June, 1989, pp. 1561-1566.

[130]Katz, A., "Special Rotation Vectors: A Means for Transmitting Quaternions in Three Components," *Journal of Aircraft*, Vol. 30, No. 1, 1993, pp. 148-150.

[131]Lo, S. C., and Y. P. Chen, "Smooth Sliding Mode Control for Spacecraft Attitude Tracking Maneuvers," *Journal of Guidance, Control, and Dynamics*, Vol. 18, No. 6, 1995, pp. 1345-1349.

[132]Joshi, S. M., Kelkar, A. G., & Went, J. T., "Robust Attitude Stabilization of Spacecraft Using Nonlinear Quaternion Feedback," *IEEE Transactions on Automatic Control*, Vol. 40, No. 10, 1995, pp. 1800-1803.

[133]Crassiidis, J. L., and Markely, F. L., "Sliding Mode Control Using Modified Rodriguez Parameters," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 6, 1996, pp. 1381-1383.

[134]Chung, D., Lee, J. G., and Park, C. G, "Strapdown INS Error Model for Multiposition Alignment," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 32, No. 4, 1996, pp. 1362-1366.

[135]Bar-Itzhack, I. Y., "REQUEST: A Recursive QUEST Algorithm for Sequential Attitude Determination," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 5, 1996, pp. 1034-1038.

[136]Kelkar, A. G., and Joshi, S. M., "Global Stabilization of Flexible Multibody Spacecraft Using Quaternion-Based Nonlinear Control Law," *Journal of Guidance, Control, and Dynamics*, Vol. 19, No. 5, 1996, pp. 1186-1188.

[137]Lee, K. L., Lee, J. G., Roh, Y. K., and Park, C. G., "Modeling Quaternion Errors in SDINS: Computer Frame Approach," *IEEE Transactions on Aerospace & Electronic Systems*, Vol. 34, No. 1, 1998, pp. 289-299.

[138]Reynolds, R. G., "Quaternion Parameterization and a Simple Algorithm for Global Attitude Estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 4, 1998, pp. 669-671.

[139]Gupta, S., "Linear Quaternion Equations with Application to Spacecraft Attitude Propagation," *IEEE Aerospace Applications Conference Proceedings*, Vol. 1, IEEE, Snow Mass, CO, March 21-28, 1998, pp. 69-76.

[140]Yu, M. J., Lee, J. G, and Park, H. W., "Comparison of SDINS In-flight Alignment Using Equivalent Error Models." *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 35, No. 3, 1999, pp. 1046-1054.

[141]Fallon, L., "Quaternions," in *Spacecraft Attitude Determination and Control*, J. R. Wertz, editor, Kluwer Academic Publishers, Dordrecht, 1978, pp. 758-759.

[142]Jones, D. W., "Quaternions Quickly Transform Coordinates Without Error Buildup," *EDN*, Vol. 40, No. 5, 1995, pp. 95-100.

[143]Bar-Itzhack, I. Y., and Idan, M., "Recursive Attitude Determination from Vector Observations: Euler Angle Estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 10, No. 2, 1987, pp. 152-157.

[144]Dvornychenko, V. N., "The Number of Multiplications Required to Chain Coordinate Transformations," *Journal of Guidance, Control, and Dynamics*, Vol. 8, No. 1, 1985, pp. 157-159.

[145]Winograd, S., "On the Number of Multiplications Necessary to Computer Certain Functions," *Communications on Pure and Applied Mathematics*, Vol. 23, No. 2, 1970, pp. 165-179.

[146]Winograd, S., "Some Bilinear Forms Whose Multiplicative Complexity Depends on the Field of Constants," *Mathematical Systems Theory*, Vol. 10, No. 2, 1977, pp. 169-180.

[147]Wehage, R. A., "Quaternions & Euler Parameters – A Brief Exposition," *Computer-Aided Analysis and Optimization of Mechanical System Dynamics*, NATO ASI Series, Vol. F9, E. J. Haug, editor, Springer-Verlag, Heidelberg, 1984, pp. 147-180.

[148]Bar-Itzhack, I. Y., "Iterative Computation of Initial Quaternion," *Journal of Spacecraft and Rockets*, Vol. 7, No. 3, 1970, pp. 367-369.

[149]Grubin, C., "Derivation of the Quaternion Scheme via the Euler Axis and Angle," *Journal of Spacecraft and Rockets*, Vol. 7, No. 10, 1970, pp. 1261-1263.

[150]Klumpp, A. R., "Reply to Comment on 'Singularity Extraction of a Quaternion from a Direction-Cosine Matrix," *Journal of Spacecraft and Rockets*, Vol. 15, No. 4, 1978, pp. 256-256.

[151]Grubin, C., "Quaternion Singularity Revisited," *Journal of Guidance and Control*, Vol. 2, No. 3, 1979, pp. 255-256.

[152]Salguero, D., "Trajectory Analysis and Optimization Software," SAND99-0811, Sandia National Laboratories, Albuquerque, NM, May 1999.

[153]Prewitt, N. C., Belk, D. M., and Maple, R. C., "Multiple Body Trajectory Calculations Using the Beggar Code," AIAA Paper 96-3384, July 1996.

[154]Thorne, W. J., "Quaternion Utilization in a Missile Model," AFIT-GST-MA-80M-3, March 1980.

[155]Kendra, M. J., and Bonito, N. A., "Coordinate System Conversion Software User's Guide for Attitude, Quaternion, M50 and Related Applications," GL-TR-90-0327, Nov. 1990.

[156]Hankey, W. L., Miller, L. E., and Scherr, S. J., "Use of Quaternions in Flight Mechanics," AFWAL-TR-84-3045, March 1984.

[157]"Guide for the Verification and Validation of Computational Fluid Dynamics Simulations," AIAA, G-077-1998, Jan. 1998.

[158]Roache, P. J., *Verification and Validation in Computational Sciences and Engineering*, Hermosa Publishers, Albuquerque, NM, 1998, pp. 1-648.

[159]Gerald, C. F., and Wheatley, P. O., *Applied Numerical Analysis*, 6th ed., Addison Wesley Longman, Reading, MA, 1999, pp. 451-477.

[160]Hoffman, J. D., *Numerical Methods for Engineers & Scientist*, McGraw-Hill, New York, 1992, pp. 225-266.

[161]Fang, A. C. and Zimmerman, B. G., "Digital Simulation of Rotational Kinematics," NASA TN D-5302, Oct. 1969.

[162]Bar-Itzhack, I. Y., "Optimum Normalization of a Computed Quaternion of Rotation," *IEEE Transactions of Aerospace and Electronic Systems*, Vol. 7, No. 2, 1971, pp. 401-402.

[163]Katz, A., *Computational Rigid Vehicle Dynamics*, Krieger, Malabar, FL, 1997, pp. 29-38.

[164]"Atmospheric and Space Flight Vehicle Coordinate Systems," ANSI/AIAA R-004-1992, Feb. 1992.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37260

AIAA-2000-4303

A GENERIC LANDING GEAR DYNAMICS MODEL FOR LASRS++

W. A. Ragsdale*
Unisys Corporation, NASA Langley Research Center, Hampton, VA 23681-2199

Abstract

The Langley Standard Real-time Simulation in C++ (LaSRS++) Generic Gear Model is a simple flexible design that provides realism and can easily simulate different aircraft by changing a few design parameters. Typical closed loop characteristics of the entire aircraft are used to model the dynamics during ground operations. The model consists of three principal functions: struts, braking, and steering. Dynamic characteristics are assumed to produce nearly critical damping with a fixed natural frequency in all axes. The model outputs the body frame axial, lateral, and vertical forces and roll, pitch, and yaw moments generated by the landing gear. This model is used in the simulation of the HL-20 lifting body, general aviation aircraft and a generic fighter similar to an F-16 at NASA Langley Research Center, including motion base operations.

Introduction

The Langley Standard Real-time Simulation in C++ (LaSRS++) is an object-oriented framework used to simulate a variety of aircraft, including general aviation aircraft, fighters, civil transports, and spacecraft. The goal of the LaSRS++ Generic Gear Model is to produce a simple flexible design that appears to operate realistically and can easily simulate different aircraft by changing a few design parameters.

The usual approach for simulating aircraft landing gear is to have detailed component and geometry models that produce open loop dynamics. This level of detail is needed when the objective of the simulation is to evaluate landing gear performance. But sometimes the data for a detailed model is not available and often a simple model that looks correct to a pilot is all that is needed.

In this design typical closed loop characteristics of the entire aircraft dynamics are used to model the contribution of landing gear forces and moments during ground operations. This results in a simple flexible design that appears to operate realistically to a pilot. Changing the design values can easily simulate different aircraft.

The generic landing gear model assumes the aircraft has tricycle landing gear with two main wheels, including brakes, and a steerable nose wheel, as shown in Figure 1. Constant values required by the model are the weight and moments of inertia of the aircraft, the body frame location of the center of gravity and wheel-ground contact points. The body frame used for the Generic Gear Model is defined with the origin at the reference center of gravity (CG), the X-axis forward, the Y-axis to the right, and the Z-axis down toward the floor of the aircraft.

Dynamic inputs to the model include aircraft attitude and rotational rates, the body frame velocity vector of the center of gravity, and body frame forces and moments produced by aerodynamics and propulsion. Pilot inputs include brake pedal inputs for left and right main gear brakes and rudder pedal inputs for nose wheel steering. Dynamic characteristics are assumed to produce nearly critical damping with a fixed natural frequency in all axes. The model outputs body frame axial, lateral, and vertical forces and roll, pitch, and yaw moments generated by the landing gear.

The model consists of three principal functions: struts, braking, and steering. The strut model produces vertical normal forces with square law damping, including the effects of stroke length limiting impact G's and the touchdown sink rate.

---

* Software Engineer, Associate Fellow, AIAA

Copyright ©2000 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S.Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

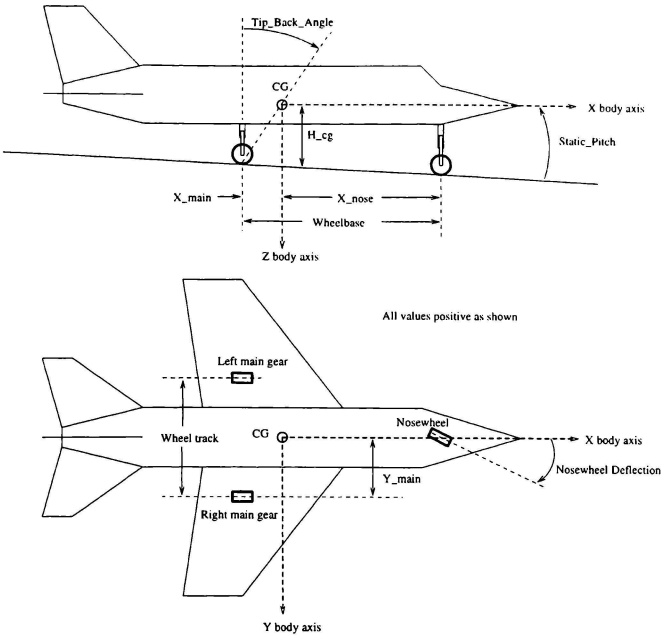


Figure 1. Generic Landing Gear Geometry

The pitching moment arm of the main gears is based on a linearized relation between the tip back angle and the static attitude of the aircraft. Braking axial forces assume a fixed rolling friction coefficient and a maximum braking coefficient of friction. Gear geometry is used to produce differential braking yaw moments and side forces are limited to simulate skidding. Axial and lateral forces decay as the aircraft stops. Nose wheel steering produces lateral forces and yaw moments based on a simple 'tricycle approximation' that avoids the traditional problem of stopping without dividing by zero.

Model Outputs

The LaSRS++ equations of motion calculate accelerations based on force and moment inputs, regardless of the state of the simulation. The Generic Gear class must generate body frame axial, lateral and vertical forces, and roll, pitch and yaw moments and provide accessor functions for other classes to use them. Output values must be updated at the iteration rate of the simulation, which for LaSRS++ is 32 iterations per second or higher. Note that this design approach precludes stopping the aircraft by forcing zero velocity inside the equations of motion. The Generic Gear Model must generate forces and moments that stop the aircraft and hold it in position when it has stopped. Also the

landing gear model must function during reset, trim and hold modes as well as in operate.

Model Constants

To use the Generic Gear Model it is necessary to supply data about the inertia and geometry of the aircraft being simulated. These values are assumed to be constant for a particular aircraft. The aircraft specifications include the landing weight of the aircraft and its roll, pitch, and yaw moments of inertia. The body frame coordinates of the center of gravity and the wheel-ground contact points with the aircraft stopped must be provided. The maximum stroke of the struts or the static pitch angle must also be specified.

The dynamic characteristics are assumed to be the same for all aircraft. The forces produced by the landing gear are generally proportional to the weight of the aircraft. The rolling, cornering, and braking coefficients of friction are independent of the aircraft being simulated. This allows the use of normalized accelerations, thus many of the gains are the same for any aircraft. All aircraft exhibit the same basic rotational characteristics during ground handling--they all have nearly critical damping in all axes and come to rest with wings level at a predictable pitch attitude and altitude of the center of gravity above the ground.

The Generic Gear Model provides default values for all of these model constants. Values can be varied by overloading the constructor function that creates the Generic Gear Model. The values can also be adjusted in real-time by a graphical user interface (GUI) that provides values for mutator functions.

All integrators and filters in LaSRS++ are required to be independent of the iteration rate. The time step is specified in the class constructor.

Variable Inputs

The Generic Gear class calculates the contribution of the landing gear forces and moments to the vehicle equations of motion. However the overall dynamics depend on the total forces and moments, such as those due to gravity, aerodynamics and propulsion. Basically the approach is to determine the total forces and moments that produce the desired dynamics, subtract the effects of gravity, aerodynamics and propulsion, and resolve what is left as the landing gear contribution. Thus the Generic Gear Model is a closed loop approach, fundamentally different from the usual open-loop simulation of ground handling.

To do these calculations the Generic Gear class requires inputs from other classes at the iteration rate of the simulation. The variable quantities that must be input include:

- Roll and pitch attitude and rates
- Yaw (heading) and track angle and rates
- Altitude of the center of gravity above the runway
- Body frame velocity vector
- Body frame forces and moments produced by aerodynamics and propulsion
- Pilot's brake input for left and right main gears
- Pilot's pedal input for nose wheel steering angle

Strut Dynamics

The struts produce vertical (or normal) forces that absorb the initial impact of the landing gear on the ground, bring the aircraft to an equilibrium attitude in pitch and roll, and then support the aircraft while rolling, braking, turning and taxiing. When the aircraft completely stops, the struts produce forces and moments that maintain the aircraft wings level with the nose at a predictable pitch attitude and the center of gravity at a predictable height above the surface. All the other forces and moments produced by the landing gear depend on the vertical or normal force produced by the struts.

In flight before landing the strut is extended to its maximum deflection, as shown in Figure 2. The initial ground impact is referred to as touchdown or 'weight on wheels'. Typically struts generate a force proportional to the square of the sink rate plus a force that depends on the deflection of the strut. This is called 'square law damping' and results in a non-linear second order differential equation. As a rule the strut is slightly underdamped--it overshoots the equilibrium deflection and oscillates at least once. The strut is usually designed to absorb a sink rate of about 10 ft/sec by generating an upward force that produces about 4 G's ('load-factor') as the strut reaches maximum deflection. More gentle sink rates produce G's proportional to the square of the sink rate. Harder touchdowns would result in crashing a real

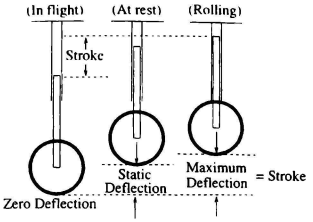


Figure 2. Strut Terminology

aircraft. The limit sink rate and load-factor in G's are specified in the Generic Gear constructor constants as described above. According to Raymer [1] the same strut deflection is required for the same sink rate and load-factor on an ultra-light as for an airliner. This is because the normal forces are generally proportional to weight and thus so is acceleration.

When both main gears are on the ground the aircraft rolls to a wings-level attitude very rapidly. The rolling motion is also an underdamped oscillation with small overshoot that damps quickly.

Because the main landing gears are behind the center of gravity they usually produce a nose-down or 'slapdown' pitching moment proportional to the normal force. But the pitch moment arm of the main gear from the center of gravity increases as the aircraft pitches down. The pitch attitude at which the center of gravity is directly above the main gear contact point is called the 'tip back angle'. If the aircraft is pitched higher while rolling forward the landing gear will pitch the nose up instead of down. In the Generic Gear Model a linearized variation of the main gear pitch moment arm depends on the pitch attitude versus the tip back angle.

When the nose gear touches down ('weight on nose gear') its strut also produces a square law underdamped oscillation. This results in the aircraft moving to an equilibrium pitch attitude. The equilibrium pitch attitude will vary depending on braking and aerodynamic pitching moments. The nose gear strut (as the mains) can push the aircraft upwards, but cannot pull it down. When the aircraft stops moving the pitch will assume a constant predictable value.

Braking Dynamics

The wheels produce axial forces due to friction acting on the wheels. The normal force and the coefficient of friction determine the axial forces. The coefficient of friction is the combination of rolling and braking friction, and static friction when the aircraft is stopped. The rolling coefficient of friction is assumed to be constant with a default value of 0.02.

Various references do not agree on the details of how to simulate braking friction. [2,3,4,5] For small brake inputs on a dry runway, the coefficient of braking friction is proportional to brake pedal deflection. The maximum braking coefficient depends on the runway condition (dry, wet, or icy) and the speed of the aircraft. Some models make the braking coefficient a function of the 'slip ratio' or ratio of the rotating wheel speed compared to the speed at the non-rotating center of the hub. The braking coefficient is reduced if the wheel stops rotating.

The approach taken in the Generic Gear Model is to make the braking friction coefficient proportional to brake pedal deflection until the maximum value is reached, and then maintain that maximum value, as indicated in Figure 3. This is effectively the same as 'anti-lock' braking. The default value for maximum braking coefficient is 0.6. This can be modified in real-time to simulate wet or icy runways if desired. The variation of maximum braking with forward speed is not simulated because it is unusual to apply maximum braking at very high speeds where it makes a difference.

If the left and right brakes are not equally applied, known as 'differential braking', a yawing moment is generated that is the product of the axial force and the lateral moment arm of the main gear tire. This moment is added to the other yawing moments, including the nose wheel steering moment as described below.

If the main tires are not pointed in the direction they are rolling, a skid angle results. This skid angle produces a side force on the tire tending to reduce the skid angle. For small skid angles the side force friction coefficient is proportional to the skid angle. But the side force or 'cornering' coefficient is limited depending on the runway condition, the same as for the braking coefficient. The side force is also limited by the amount of braking being applied, partly because

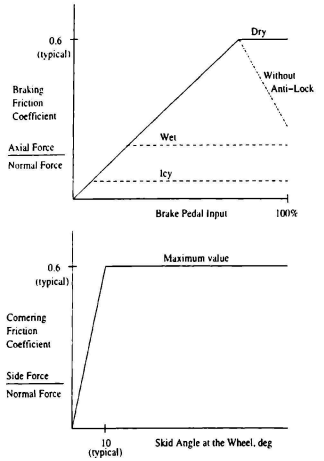


Figure 3. Friction Coefficients

the root-sum-square magnitude of the two friction coefficients is physically limited as well.

Probably the most difficult ground dynamics phenomenon to simulate in real-time is stopping the aircraft. When the aircraft stops the friction forces disappear, and the aircraft cannot be moved or turned by using the brakes or nose wheel while it is stopped. If the usual equations are applied continuously the simulated aircraft backs up or turns unrealistically when stopping. Several ground dynamics functions, especially the skid angle, depend on dividing by the speed of the aircraft. If the speed is allowed to go to exactly zero the program will crash due to a 'divide by zero' error. Even if speed only decays to a small number, unrealistic seemingly random results occur.

The classical way to solve this problem is to set the velocity directly to zero inside the equations of motion when the velocity approaches zero. Since the LaSRS++ architecture does not allow the landing gear model to modify the velocity of the aircraft, it can only output forces and moments that are integrated to produce translational and rotational velocities.

In the LaSRS++ Generic Gear Model the axial and lateral friction forces are decayed to zero by making them proportional to forward and lateral velocity at low speeds. This prevents the aircraft from backing up, sliding sideways, or turning after it stops. If there are crosswinds producing aerodynamic side forces, the gear model will produce enough side force to keep the aircraft static, up to the limit of side force friction. Only when the forward thrust of the engines exceeds the static friction will the aircraft start to move again.

Steering Dynamics

The turn rate of the aircraft depends on differential braking, the nose wheel steering angle, and the ground-relative speed of the aircraft. The mathematical relationship between these quantities is the same for all aircraft. Both the main gear and nose gear generate side forces when the tires are not aligned with the direction they are rolling, producing a skid angle. Most ground dynamics models calculate the side forces on all tires depending primarily on the skid angle at the wheel. When the nose wheel is deflected left or right from the fuselage centerline the resulting local skid angle produces a side force and yawing moment that causes the aircraft to turn. The major problem with this open loop approach is that the skid angle becomes very large as the forward velocity decays to zero, causing unrealistic forces and oscillations.

The Generic Gear Model simultaneously solves the problems of nose wheel steering and stopping by using what is called the 'tricycle approximation', as shown in Figure 4. This method resulted from the observation that while turning on the ground the radius of curvature of the turn is always near the intersection of the extended axle lines of the main gear and nose gear. This allows a simple approximation that predicts a reference rate of turn proportional to both the nose wheel steering angle and the forward velocity of the aircraft. The total side or 'centripetal' force to maintain the turn is based on elementary physics. The total yawing moment is proportional to the difference between the reference turn rate and the aircraft yaw rate. The tendency of an unrestrained nose wheel to align with the local velocity vector ('castering') is simulated by adjusting the yaw rate gain. The side force on the nose gear is also limited by the maximum friction coefficient similar to the main wheel model described above.

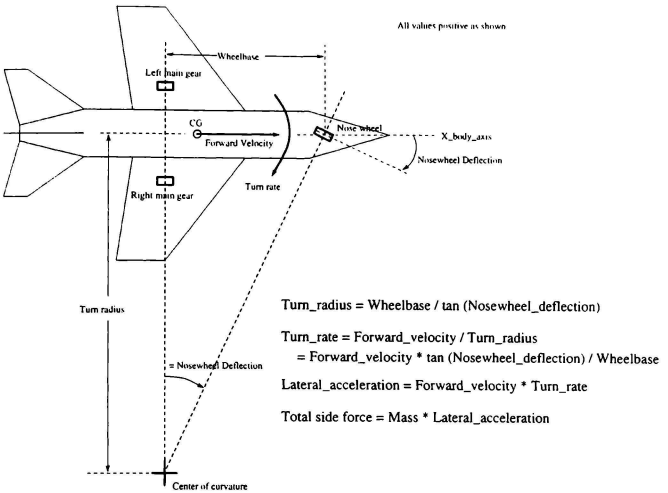


Figure 4. The Tricycle Approximation

The Generic Gear Model outputs the required total side force and yawing moment to sustain the turn. Although it is several orders of magnitude simpler, this method produces nearly the same results as the more complex models in the references, [3,6] and is smoother at low speeds.

Results

The current version of the real-time update function of the Generic Gear Model that is executed in real-time operate mode consists of a dozen logical decisions and about 30 assignment statements.

The first real-time simulation at Langley to use the LaSRS++ Generic Gear Model was the HL-20 lifting body in August 1999. An earlier version of the model, coded in FORTRAN, was used starting in 1990, because no landing gear model was available for the HL-20. The HL-20 was not primarily concerned with ground dynamics, but pilots were interested in seeing if the vehicle could be stopped in the remaining runway after touchdown. When the simulation was flown on a motion base it was discovered that the 'slapdown' immediately after ground contact was more severe than expected, because the elevons sometimes went to their upper limits after touchdown. Moving the main landing gear forward to reduce the main gear pitch moment arm solved the problem. This project also investigated the limits for crabbed crosswind landings, since it is tricky to decrab a lifting body landing deadstick at 210 knots. The gear model provided preliminary estimates of the required strength of the landing gear and control authority during landing and rollout.

The 'tricycle approximation' was first tested with the Rollout and Turnoff (ROTO) project modeling a Boeing 737 in FORTRAN in 1997. One of the objectives of this project was to simulate high-speed turnoffs requiring high gains on the motion base. The discontinuities in the landing gear model at low speed became very pronounced on the motion base. The tricycle approximation was used at low speeds to

alleviate these discontinuities and still produce realistic results.

The LaSRS++ General Aviation Baseline project is designed to easily simulate different aircraft by changing only a few program parameters, including aerodynamics, propulsion, and landing gear. The Generic Gear Model was used for this simulation since no landing gear model was specified and the model already provides the needed flexibility. The same object-oriented class used for the HL-20 is the base class for the General Aviation Baseline gear class. Its constructor function provides default values applicable to the aircraft being simulated. Values can be changed in real-time via a graphical user interface (GUI). This project has been simulating a Cessna 172 aircraft, including takeoff, landing, and taxiing, since October 1999 in a fixed-base cockpit.

The LaSRS++ generic fighter simulation models an aircraft similar to an F-16 and is used as a test bed for the LaSRS++ framework. It includes the Generic Gear Model and demonstrates the ability to model ground dynamics for such aircraft by changing only the gear model inertia and geometry parameters. This provided more confidence in the typical dynamics parameters that are assumed to be the same for all aircraft.

Figures 5(a) and 5(b) show the results of an HL-20 landing using the Generic Gear Model. The run begins with the aircraft 5 feet above the ground with a sink rate about 4 feet/sec. The following control inputs were made:

- At 5 seconds 50% braking is applied.
- At 15 seconds differential braking, 60% left and 40% right, is applied
- At 20 seconds brakes are released
- At 25 seconds 50% right rudder pedal is applied, deflecting the nose wheel 7.5 degrees
- At 30 seconds the pedals are centered
- At 35 seconds 50% left rudder pedal is applied
- At 40 seconds the pedals are centered and 25% brakes are applied until the aircraft stopped at around 45 seconds

These plots were generated by a batch test, using step inputs that are more abrupt than with a pilot in the loop. The damping and frequency of the dynamics are evident.

Conclusions

The LaSRS++ Generic Gear Model has demonstrated a simple and flexible model that can be used to simulate ground dynamics for a variety of aircraft, including general aviation, fighters, and lifting bodies. The Generic Gear Model is presently being used in a simulation of general aviation, the HL-20 lifting body, and a generic fighter similar to an F-16. There are also plans to use the Generic Gear Model in a future civil transport simulation. The results are not notably different from those produced by more complex models, and in some cases are more realistic, especially at low speeds.

References

1. Raymer, Daniel P., "Aircraft Design: A Conceptual Approach," AIAA Education Series, 1989

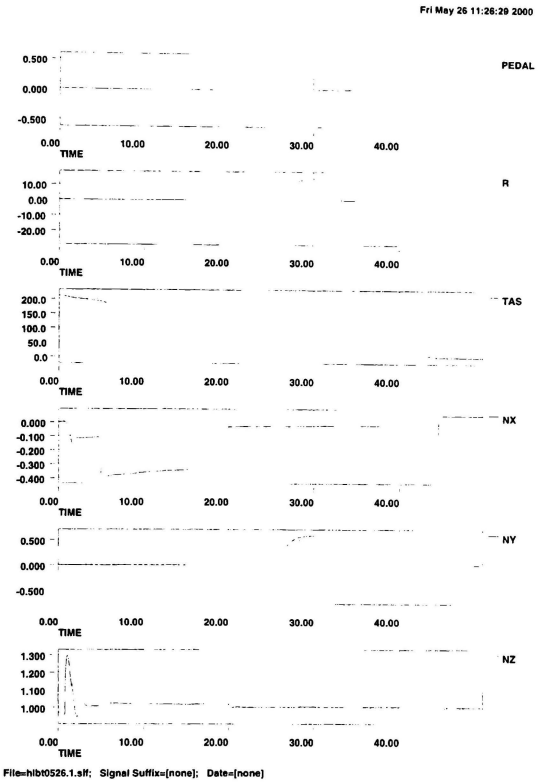
2. Hanke, C. Rodney, "The Simulation of a Large Jet Transport Aircraft Volume I: Mathematical Model," The Boeing Company, NASA CR-1756, March 1971

3. Passmore, Harry and Korba, C.R., "DC-9 Landing Gear Math Model for Directional Control on Runway Flight Simulation," McDonnell Aircraft Company, Report Number MDC A4816, May 1977

4. Karp, Rein, "General Aviation Math Model," Sperry Support Services, NASA Langley Research Center, Report Number SP-710-0006, July 1977

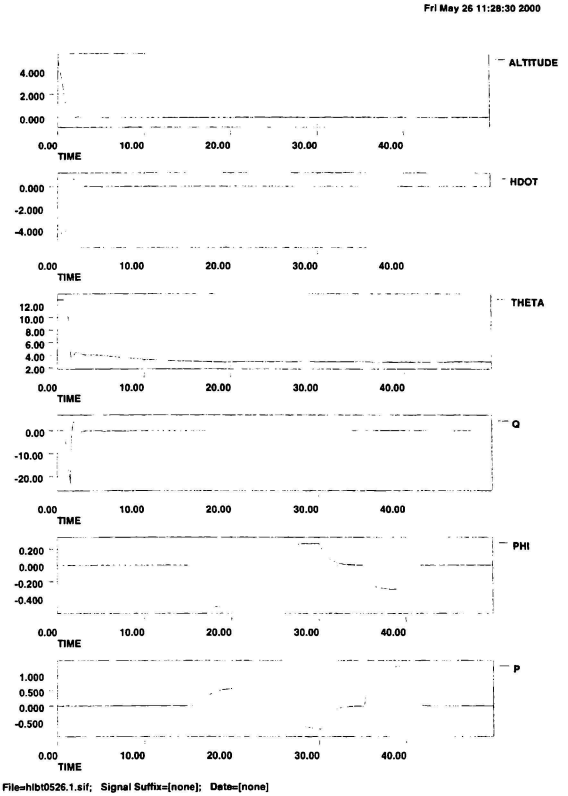
5. Wahi, M.K. and Yourkowski, F.M., "757-200 Training Simulator - Ground Handling Data, The Boeing Company, Document Number D613N104, July 1990

6. Pines, S., Schmidt, S.F. and Mann, F., "Simulation, Guidance and Navigation of the B-737 for Rollout and Turnoff Using MLS Measurements," Analytical Mechanics Associates, Inc., NASA Langley Research Center, AMA Report Number 75-40, Dec 1975



t=5 sec 50% brakes, t=15 sec 60/40% brakes, t=20 sec zero brakes, t=25 sec 50% pedal, t=30 sec zero pedal, t=35 sec -50% pedal, t=40 sec zero pedal & 25% brakes, t=45 stop

Figure 5(a). HL-20 Landing Dynamics Using the Generic Gear Model



t=5 sec 50% brakes, t=15 sec 60/40% brakes, t=20 sec zero brakes, t=25 sec 50% pedal, t=30 sec zero pedal, t=35 sec -50% pedal, t=40 sec zero pedal & 25% brakes, t=45 stop

Figure 5(b). HL-20 Landing Dynamics Using the Generic Gear Model

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37261

AIAA-2000-4304

AN AERODYNAMIC MODEL FOR SIMULATION OF CLOSE FORMATION FLIGHT

William B. Blake
Air Force Research Laboratory
Wright Patterson AFB OH 45433-7531

ABSTRACT

A mathematical representation of the aerodynamics for simulation of a large number of vehicles in close formation flight is presented. Aerodynamic coupling effects between the aircraft due to wake interference are included. A combination of wind tunnel results and vortex lattice analysis is used to significantly reduce the number of state variables included in the aerodynamic coupling terms. Computational results are presented for two tailless aircraft at a wide variety of relative locations.

NOMENCLATURE

b	Wing span
c	Wing mean aerodynamic chord
C_D	Drag coefficient
C_L	Lift coefficient
C_l	Rolling moment coefficient
C_m	Pitching moment coefficient
C_n	Yawing moment coefficient
C_Y	Side force coefficient
C_Z	Vertical force coefficient
h	Height above ground
M	Mach number
p	Body axis roll rate
q	Body axis pitch rate
r	Body axis yaw rate
V	Velocity
x_w	Longitudinal distance to aircraft wake
y_w	Lateral distance to aircraft wake
z_w	Vertical distance to aircraft wake
α	Angle of attack
β	Angle of sideslip
δ	Control deflection
ϕ_w	Roll inclination of aircraft relative to wake
θ_w	Pitch inclination of aircraft relative to wake
ψ_w	Yaw inclination of aircraft relative to wake

* Technical Area Leader, Associate Fellow AIAA

This paper is declared a work of the U.S. government and is not subject to copyright protection in the United States.

INTRODUCTION

Military aviation contains many examples of aircraft formation flight. One type of formation flight is close formation flight, where there is significant aerodynamic coupling between aircraft. This coupling arises from aircraft flying within wakes generated by other aircraft in the formation. Close formation flight is avoided in civilian aviation due to safety concerns. Military applications of close formation flight include aerial refueling and minimum interval takeoff. If the aircraft are appropriately positioned, upwash from wing tip vortices can reduce the drag of neighboring aircraft, increasing range or endurance [1,2]. This is used by flocks of migratory birds. Aircraft range increases of 50% or more are believed to be attainable, with larger benefits obtained from larger formations. Study of this idea requires the simulation of a large number of aircraft, each affected by the presence of the others. The objective of the current work is to develop a math model suitable for this task.

Real time simulations of a single aircraft model the vehicle aerodynamics as functions of about ten state variables. Linear derivatives or table-lookups are used to compute the variation of the aerodynamic parameters with the aircraft states. Multi-dimensional tables are often used, with the number of states kept to a minimum to keep the computational requirements low. Even so, a supersonic fighter aerodynamic model can contain over 500,000 data points [3]. One method for modeling the aerodynamics of the trail aircraft in a

two ship formation would be to add all of the states necessary to model the lead vehicle plus six additional ones representing the relative positions and orientations of the two vehicles. The size of such a simulation model would be extremely large, and extension to formations of three or more aircraft would be prohibitive. To simulate a large formation of aircraft, significant reductions in the number of states must be made.

Many simulations of a single aircraft flying within the wake of another have been conducted [4-9]. These have studied the safety impact of wake encounters during approach and landing operations or supported accident investigations where an out of control situation due to wake encounter was a possible cause. Only a handful of additional states are included in the representation of the trail aircraft, with the wake typically held fixed in inertial space. Effects of the trail aircraft on the wake generating aircraft or the wake itself are ignored. The present work develops a model for an arbitrarily large formation of tailless aircraft.

DISCUSSION

The six-degree-of-freedom equations of motion for a rigid body can be written as:

$$m(\dot{u}+qw-rv)=QSC_X-mg\sin\theta \quad [1]$$

$$m(\dot{v}+ru-pw)=QSC_Y+mg\cos\theta\sin\phi \quad [2]$$

$$m(\dot{w}+pv-qu)=QSC_Z+mg\cos\theta\cos\phi \quad [3]$$

$$I_x\dot{p}+qr(I_z-I_y)-(\dot{r}+pq)I_{xz}=QSbC_l \quad [4]$$

$$I_y\dot{q}-pr(I_z-I_x)+(p^2-r^2)I_{xz}=QScC_m \quad [5]$$

$$I_z\dot{r}+pq(I_y-I_x)+(qr-\dot{p})I_{xz}=QSbC_n \quad [6]$$

Here, p, q and r are the angular rates about, and u, v and w are the velocity components along the body fixed X, Y and Z axes respectively. Aerodynamic forces and are given in non-dimensional form with Q as the free-stream dynamic pressure, S as the reference area, and b and c as reference lengths. The forces along the body axes are given by C_X, C_Y and C_Z, and the moments about the axes are given by C_l, C_m and C_n. It is customary to express the force coefficients in terms of lift and drag, which are oriented relative to the free-stream. With the aircraft at angle of attack α, the body axis force coefficients become:

$$C_X=-C_D\cos\alpha+C_L\sin\alpha \quad [7]$$

$$C_Z=-C_L\cos\alpha-C_D\sin\alpha \quad [8]$$

The challenge for the aerodynamicist is to determine values for the force and moment coefficients which encompass all possible flight conditions and control settings (δ). The aerodynamic math model that results from this effort is typically a hybrid of wind tunnel measurements, analytical predictions, and empirical estimates from prior designs. High fidelity simulations represent the coefficients as functions of about ten state variables:

$$C_i=C_i\langle\alpha,\dot{\alpha},\beta,p,q,r,M,h,\delta_1,\delta_2,\ldots,\delta_n\rangle \quad [9]$$

The list is representative but not exhaustive, landing gear, external stores and other variables are sometimes added. Depending on the range of flight conditions to be simulated and the available analytical and experimental data, derivative or table look-up representations are used to model the state variable effects. For the tailless aircraft studied here, a large experimental database was available which allowed use of the following representation [3]:

$$C_{Lo}=C_L\langle\alpha,\beta,M\rangle+\sum C_L\langle\delta_k,\alpha,M\rangle+\frac{\partial C_L}{\partial(qc/2V)}\langle\alpha,M\rangle(\frac{qc}{2V}) \quad [10]$$

$$C_{Do}=C_D\langle\alpha,\beta,M\rangle+\sum C_D\langle\delta_k,\alpha,M\rangle \quad [11]$$

$$C_{mo}=C_m\langle\alpha,\beta,M\rangle+\sum C_m\langle\delta_k,\alpha,M\rangle+\frac{\partial C_m}{\partial(qc/2V)}\langle\alpha,M\rangle(\frac{qc}{2V})+\frac{\partial C_m}{\partial(\dot{\alpha}c/2V)}\langle\alpha,M\rangle(\frac{\dot{\alpha}c}{2V}) \quad [12]$$

$$C_{lo}=C_l\langle\alpha,\beta,M\rangle+\sum C_l\langle\delta_k,\alpha,M\rangle+\frac{\partial C_l}{\partial(pb/2V)}\langle\alpha,M\rangle(\frac{pb}{2V})+\frac{\partial C_l}{\partial(rb/2V)}\langle\alpha,M\rangle(\frac{rb}{2V}) \quad [13]$$

$$C_{no}=C_n\langle\alpha,\beta,M\rangle+\sum C_n\langle\delta_k,\alpha,M\rangle+\frac{\partial C_n}{\partial(pb/2V)}\langle\alpha,M\rangle(\frac{pb}{2V})+\frac{\partial C_n}{\partial(rb/2V)}\langle\alpha,M\rangle(\frac{rb}{2V}) \quad [14]$$

$$C_{Yo}=C_Y\langle\alpha,\beta,M\rangle+\sum C_Y\langle\delta_k,\alpha,M\rangle+\frac{\partial C_Y}{\partial(pb/2V)}\langle\alpha,M\rangle(\frac{pb}{2V})+\frac{\partial C_Y}{\partial(rb/2V)}\langle\alpha,M\rangle(\frac{rb}{2V}) \quad [15]$$

The subscript o refers to a single aircraft not affected by the presence of any other aircraft. The question posed for the formation flight case is what new terms and state variables are needed to adequately represent the wake induced effects.

PRIOR MATH MODELS

A large number of wake vortex encounter simulations have been conducted [4-9]. In these studies, the position of the wake is prescribed and only the motion of the trail vehicle is simulated. Most model the wake as a single pair of fully rolled up vortices. Wake induced effects are calculated using strip theory. This theory divides the wing into a number of streamwise strips, calculates the local velocity at each strip, resolves it into effective angles of attack and sideslip, computes the load on each strip, and integrates the resulting load distribution. Bernstein and Iversen [5] conducted a three degree-of-freedom simulation modeling the vortex induced effects as follows:

$$C_L = C_{Lo} + \frac{\partial C_L}{\partial \Gamma}\langle y_w, z_w \rangle \Gamma \qquad [16]$$

$$C_Y = C_{Yo} \qquad [17]$$

$$C_l = C_{lo} + \frac{\partial C_l}{\partial \Gamma}\langle y_w, z_w \rangle \Gamma \qquad [18]$$

Here, y_W and z_W represent the distance from the aircraft to the midpoint of the trailing vortex pair, measured in an axis system aligned with the vortices, as shown in Figure 1.

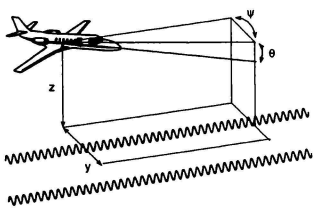


Figure 1. Aircraft-Wake Orientation

Γ is the circulation strength of the vortex system. Bernstein and Iversen numerically integrated the strip theory equations and included curve fits of the results into the simulation. For a typical aircraft, the largest wake induced effects are the lift and rolling moment, so this is a good representation for preliminary study.

Nelson [6] computed all six forces and moments using a strip analysis of the wing and included the strip theory integration in the simulation. A later series of piloted simulations [7,8] added strip theory calculations for the vortex induced loads on the horizontal and vertical tails. Written using the present nomenclature, the form of the math model used in these studies is:

$$C_L = C_{Lo} - \frac{\partial C_Z}{\partial \Gamma}\cos\alpha \langle y_w, z_w, \theta_w, \psi_w, \phi_w \rangle \Gamma \qquad [19]$$

$$C_D = C_{Do} - \frac{\partial C_Z}{\partial \Gamma}\sin\alpha \langle y_w, z_w, \theta_w, \psi_w, \phi_w \rangle \Gamma \qquad [20]$$

$$C_m = C_{mo} + \frac{\partial C_m}{\partial \Gamma}\langle y_w, z_w, \theta_w, \psi_w, \phi_w \rangle \Gamma \qquad [21]$$

$$C_l = C_{lo} + \frac{\partial C_l}{\partial \Gamma}\langle y_w, z_w, \theta_w, \psi_w, \phi_w \rangle \Gamma \qquad [22]$$

$$C_n = C_{no} + \frac{\partial C_n}{\partial \Gamma}\langle y_w, z_w, \theta_w, \psi_w, \phi_w \rangle \Gamma \qquad [23]$$

$$C_Y = C_{Yo} + \frac{\partial C_Y}{\partial \Gamma}\langle y_w, z_w, \theta_w, \psi_w, \phi_w \rangle \Gamma \qquad [24]$$

Note that only two force calculations are performed, axial force (C_X) is neglected. Lift and drag are computed from the vertical force (C_Z). The Euler angles (θ_W, ψ_W, ϕ_W) represent the relative orientation of the trailing vortices (fixed in inertial space) and the trail vehicle. Since the wake generating vehicle may have an entirely different flight path from the encountering vehicle, these angles can become quite large. Hastings and Keyser [Ref. 9] used the above form with a time varying circulation strength, $\Gamma(t)$, to model the effect of vortex aging. A powerful simplification implicit in these models is that the net effect on the trail vehicle of all of the states of the wake generating vehicle is represented by a single state, the circulation strength. For an elliptically loaded wing, it can be written as a function of the lift coefficient as follows:

$$\Gamma = \frac{4W}{\pi \rho V b} = \frac{2VS}{\pi b} C_L \qquad [25]$$

The models assume the separation between the lead and trail aircraft is large enough to neglect the effects of vortex length, i.e. dependencies in the x-direction. A large separation also allows any upwash effect from the trail vehicle to be ignored on the lead vehicle.

In the aerial refueling simulation developed by Weiss [10], a horseshoe vortex model was used, with the

vortex subdivided into three longitudinal segments, one for the initial rollup, one for a fully rolled up vortices, and a final one for vortex breakup. During rollup, the swirl velocity was linearly increased from half to full strength while during decay it was linearly reduced from full strength to zero. Increases in local velocity due to engine exhaust from the tanker were also estimated. The velocity increments were translated into angle of attack and sideslip increments, which were applied to the receiver aircraft using strip theory. Weiss also included a dynamic pressure ratio loss on the receiver aircraft due to flow separation from the tanker, using the methodology from the USAF Datcom handbook [11].

PROPOSED MATH MODEL

For a large formation of aircraft, the positions of the wakes will not be fixed but will move with each aircraft. To determine the aircraft/wake orientations, it is assumed that the wakes move downstream parallel to the velocity vector of the aircraft. To determine the aircraft/wake orientation, a coordinate system is used which is centered on the wake penetrating aircraft but aligned with the trailing vortices, as shown in Figure 2. It is rotated from the body fixed axes, from which the forces, moments and equations of motion are computed, by the Euler angles used previously. It is assumed that aircraft in formation flight follow essentially the same flight path, so the Euler angles are small.

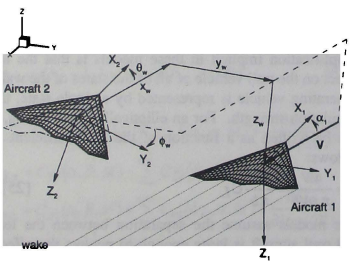


Figure 2. Aircraft Wake Orientation

Gingras [2] conducted a wind tunnel test of two aircraft in close proximity. He measured six component forces and moments on a 0.10 scale F-18 that was positioned downstream of an F-18 wing mounted on a body. The lead aircraft was fixed at ten degrees angle of attack. Positional state variables were varied to the extent allowed by the apparatus, as follows:

	$\Delta x_w/b$	$\Delta y_w/b$	$\Delta z_w/b$	θ_w	ψ_w
min	1.5	-1.5	-0.6	0	-5
max	2.25	1.5	0.9	10	5

Table 1. Parameter Variation in Close Formation Wind Tunnel Test [Ref 2].

Wake induced effects were strongly influenced by changes in lateral and vertical position, y_w and z_w. There was insufficient variation in the longitudinal (x_w) direction to draw any firm conclusion. All of the wake induced force and moment *increments* except for drag were found to be independent of the angle of attack of the trail aircraft. Since the wake position was fixed, any effect due to trail aircraft angle of attack is equivalent to an effect due to aircraft/wake orientation (θ_w). Yaw variations were obtained by pivoting about an axis located aft of the trail aircraft. With the lead aircraft position fixed, this resulted in an additional offset in lateral position (y_w). When the data from all yaw runs were plotted together, there was no discernable effect of yaw over and above the effect of lateral position. Summarizing, the wind tunnel results indicate the state variables y_w and z_w must be included, ψ_w may be ignored, and θ_w need only be included in the drag calculation.

The vortex lattice code HASC95 [12] was used to investigate variations in angle of attack of the lead and trail aircraft and the position variables x_w, y_w, z_w, and ϕ_w. HASC95 was found to give very good predictions of the trends and fair predictions of the magnitudes of the wake induced forces and moments measured in Gingras's test. The study aircraft was a tailless fighter configuration that has been extensively wind tunnel tested and simulated [3]. A two aircraft formation was studied with nominal spacings of: $x_w/b=2$, $y_w/b=1$, $z_w/b=0$ and $\phi_w=0$. Results are shown for a nominal flight condition of M=0.8 at 40000 ft. Each aircraft was modeled with 360 panels, 36 in the spanwise direction and 10 in the chordwise direction. The spanwise panels were evenly spaced, cosine spacing was used for the chordwise elements. Care was taken to ensure correct alignment of vortex filaments and control points when the aircraft overlapped in the lateral direction. The HASC95 representation of the two aircraft is shown in Figure 2.

The predicted wake induced increments to the six forces and moments at the nominal spacing are shown in figure 3, as a function of the lift coefficient of the lead aircraft. To better present all of the results on a single plot, the incremental lift coefficient will henceforth be shown reduced by a factor of three. The angles of attack of the aircraft were identical and varied from zero to ten degrees. Runs were made with the lead aircraft angle of attack varied and the trail aircraft angle of attack fixed, and vice versa. The results with respect to variation in lift coefficient were identical for all three cases. The wake induced lift, rolling moment and pitching moment vary linearly with the lift of the lead aircraft and are not affected by the lift of the trail aircraft (or θ_W). This is consistent with the F-18 wind tunnel results. The wake induced drag, yawing moment and side force vary in a parabolic fashion. Only the F-18 drag measurements showed non-linear variations with aircraft/wake orientation. The yawing moment and side force increments measured on the F-18 were presumed to be dominated by wake induced sidewash at the vertical tail.

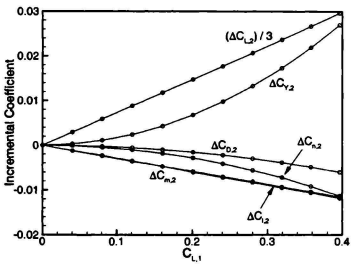


Figure 3. Effect of lead aircraft lift on trail aircraft forces and moments, x_W=2, y_W=1, z_W=0.

For the present all wing configuration, the yaw and side force increments are dominated by differences in drag across the wing span. This would make these parameters functions of the angle of attack of the wake penetrating aircraft (or θ_W for a fixed wake). This is consistent with a horseshoe vortex analysis of a wing alone [1]. That analysis showed the drag and yawing moment vary linearly with the product of the lift coefficient and of the lead and trail aircraft. Figure 4 shows the variation in wake induced drag, yawing moment and side force with the lift coefficient product. Linear variations are found.

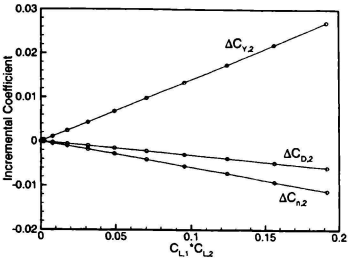


Figure 4. Effect of product of lead and trail aircraft lift on trail aircraft drag, side force and yawing moment, x_W=2, y_W=1, z_W=0.

The effect of relative lateral position on the wake induced increments is shown in Figure 5 for the nominal spacing. Nonlinear behavior is evident for all of the increments. With zero roll offset, the results are either symmetric or anti-symmetric about y_W axis. This reduces the size of the data tables necessary for table look-up. The magnitudes of the predicted increments are small. By comparing them to those obtained from a control deflection, we can assess if they will have an appreciable effect on the aircraft motion. Increments available from various controls on the left wing at the reference condition [3] are shown in Table 2.

Control	ΔC_m	ΔC_l	ΔC_n
Elevon (δ=30)	-0.0278	0.0139	-0.0034
(δ=-30)	0.0282	-0.0142	-0.0049
Wingtip (δ=30)	0.0195	0.0098	-0.0103
(δ=-30)	-0.0077	-0.0166	-0.0053
Clamshell (δ=60)	-0.0007	-0.0002	0.0068
Yaw thrust vector	0	0	-0.0042

Table 2. Aerodynamic Increments due to Control Deflection on Left Wing

A positive deflection is trailing edge down. Equal or opposite increments are obtained for right wing control deflections. Also shown are the increments available from thrust vectoring, which are small at this flight condition due to the low engine thrust at 40000 ft. Comparison with the values shown in Figure 5 shows

all of the wake induced increments are significant. When the lead aircraft is directly behind the trail ($y_W/b=0$), the lift, drag and pitching moment increments reach their peak values. Here, the trail aircraft is immersed in the downwash of the lead. The incremental pitching moment is about half of that available from the elevon. Incremental rolling moment, yawing moment and side force reach peak values at about 1/2 span spacing, which corresponds to the nose of the trail aircraft directly behind the wing tip of the lead. These increments are roughly equivalent to that available from a single control, so a combination of devices must be used for trim. For the majority of positions, the induced rolling and yawing moments are of the same sign. This results in a coordinated motion, requiring a control device with similar characteristics to counter.

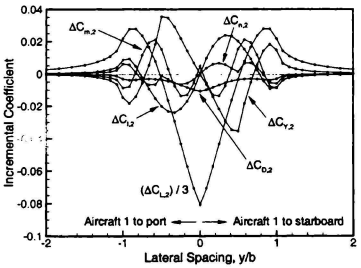


Figure 5. Effect of lateral spacing on trail aircraft forces and moments, $x_W=2$, $z_W=0$.

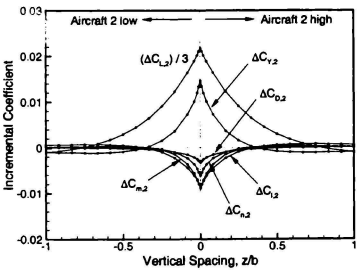


Figure 6. Effect of vertical spacing on trail aircraft forces and moments, $x_W=2$, $y_W=1$.

The effect of relative vertical position on the wake induced increments is shown in Figure 6 for the nominal spacing. In all cases, the peak values occur with co-planar aircraft with sharp reductions as the aircraft move out of plane in either direction. Symmetrical results are obtained since the aircraft was modeled with no camber. As with the variations in the lateral direction, this reduces the size of the data tables necessary for table look-up. Results from Gingras's test support the cusp in the curves found at $z_W/b=0$, as indicated in Figure 7. Good agreement is found between the HASC95 predictions and the test data.

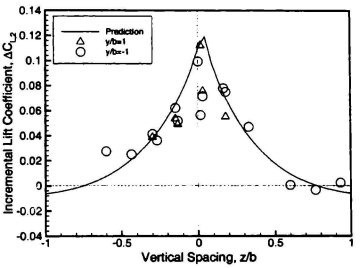


Figure 7. Measured and predicted effect of vertical spacing on trail aircraft lift (F-18), $x_W=2.25$, $y_W=1$.

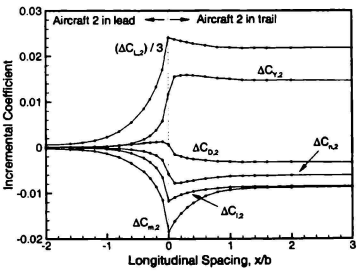


Figure 8. Effect of longitudinal spacing on trail aircraft forces and moments, $y_W=1$, $z_W=0$.

The effect of relative longitudinal position on the wake induced increments is shown in Figure 8 for the nominal spacing. Results are shown for aircraft 2 both behind and ahead of aircraft 1, the wake generating

aircraft. The influence of the wake is only significant for about one wing span upstream. Downstream, the wake influence changes rapidly and reaches its final value about two wing spans downstream. The coefficients reach their maximum magnitude at or near zero spacing, corresponding to side by side flight.

A formation shape currently in use by the USAF for long range ferry missions and airdrop operations is a single row echelon. This formation allows freedom of lateral movement without the possibility of collision, a positive safety feature that the "V" formations used in World War II did not have. A version of this formation has been proposed for close formation flight missions [1] with the lead aircraft rotating to the rear on a periodic basis. The current study aircraft has a length to span ratio of about 1.2. To allow freedom of lateral movement, the longitudinal spacing must be greater than this in both directions. Beyond this boundary, figure 8 shows very small effects of longitudinal position in both directions.

The effect of roll orientation of the trail vehicle on the wake induced increments is shown in Figure 9 for the nominal spacing. This effect is very similar to the effect of vertical spacing. The results are nearly symmetrical about zero roll, with slightly steeper gradients with the aircraft rolling away from the lead.

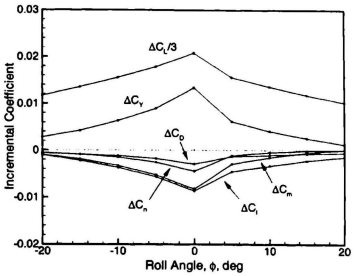


Figure 9. Effect of roll orientation on trail aircraft forces and moments, x_W=2, y_W=1, z_W=0.

To extend the results to large formations, we assume superposition of all of the individual leader/follower combinations. This results in the following form for the force and moment coefficients:

$$C_{L,i} = C_{Lo,i} + \sum_{j,i\neq j} \frac{\partial C_{L,i}}{\partial C_{L,j}} \langle x_w, y_w, z_w, \phi_w \rangle C_{L,j} \quad [26]$$

$$C_{D,i} = C_{Do,i} + \sum_{j,i\neq j} \frac{\partial C_{D,i}}{\partial C_{L,i} C_{L,j}} \langle x_w, y_w, z_w, \phi_w \rangle C_{L,i} C_{L,j} \quad [27]$$

$$C_{m,i} = C_{mo,i} + \sum_{j,i\neq j} \frac{\partial C_{m,i}}{\partial C_{L,j}} \langle x_w, y_w, z_w, \phi_w \rangle C_{L,j} \quad [28]$$

$$C_{l,i} = C_{lo,i} + \sum_{j,i\neq j} \frac{\partial C_{l,i}}{\partial C_{L,j}} \langle x_w, y_w, z_w, \phi_w \rangle C_{L,j} \quad [29]$$

$$C_{n,i} = C_{no,i} + \sum_{j,i\neq j} \frac{\partial C_{n,i}}{\partial C_{L,i} C_{L,j}} \langle x_w, y_w, z_w, \phi_w \rangle C_{L,i} C_{L,j} \quad [30]$$

$$C_{Y,i} = C_{Yo,i} + \sum_{j,i\neq j} \frac{\partial C_{Y,i}}{\partial C_{L,i} C_{L,j}} \langle x_w, y_w, z_w, \phi_w \rangle C_{L,i} C_{L,j} \quad [31]$$

The lift coefficients of the aircraft must be solved for simultaneously. The set of equations represented by Eq. 26 is solved as follows:

$$\begin{bmatrix} C_{L,1} \\ C_{L,2} \\ \dots \\ C_{L,n} \end{bmatrix} = \begin{bmatrix} 1 & -\frac{\partial C_{L,1}}{\partial C_{L,2}} & \dots & \frac{\partial C_{L,1}}{\partial C_{L,n}} \\ -\frac{\partial C_{L,2}}{\partial C_{L,1}} & 1 & \dots & -\frac{\partial C_{L,2}}{\partial C_{L,n}} \\ \dots & \dots & & \dots \\ -\frac{\partial C_{L,n}}{\partial C_{L,1}} & -\frac{\partial C_{L,n}}{\partial C_{L,2}} & \dots & 1 \end{bmatrix}^{-1} \begin{bmatrix} C_{Lo,1} \\ C_{Lo,2} \\ \dots \\ C_{Lo,n} \end{bmatrix}$$

Once the lift coefficients are computed, the remaining coefficients are easily found.

Superposition is not entirely valid for a long echelon formation since individual wakes will be modified as they encounter an aircraft. However, the relative lateral spacing will become large, and Figure 5 indicates very little effect for spacings greater than two spans. For this type of formation, one possible simplification would be to only include the wake induced effects on neighboring aircraft. This would only retain the central, upper and lower diagonal rows of the matrix in eq. (32).

Figures 5-8 can be used to find regions of positional stability, where the wake induced forces tend to return the trail aircraft to its original position when disturbed. For the sign conventions selected, there are six new positional stability derivatives:

$$\begin{aligned} &\partial C_L / \partial z_w < 0 \\ &\partial C_D / \partial x_w < 0 \\ &\partial C_m / \partial z_w < 0 \\ &\partial C_Y / \partial y_w > 0 \\ &\partial C_l / \partial y_w > 0 \\ &\partial C_n / \partial y_w > 0 \end{aligned} \qquad [33]$$

Results from HASC95 were used to generate a map of the stable regions, shown in Figure 10. A symbol within a region indicates stability. All parameters except drag are shown, it was effectively neutrally stable at "safe" following distances of > 1.5 spans downstream. There are no regions where all derivatives are stable (or unstable). While the regions of lift, pitch and roll stability will be about the same for any aircraft, regions of side force and yaw stability may depend on the presence and location of vertical tail(s) and will be configuration dependent. The symmetry in side force about the horizontal axis results from the symmetry of the present configuration (tailless) about the same axis.

These stability derivatives have been previously studied by Bloy et al [13]. They conducted analytical and wind tunnel investigations of aerial refueling, and found results similar to those shown for the trail aircraft beneath the leader ($C_{ly}>0$, $C_{my}>0$, $C_{Yy}<0$). They also studied derivatives with respect to bank and yaw angle, and found a significant destabilizing effect of the vertical tail on the yaw derivatives.

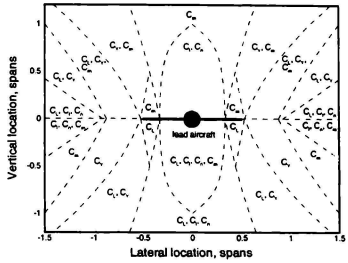


Figure 10. Regions of Positional Stability Relative to Lead Aircraft

Precision flight demonstration teams such as the USAF Thunderbirds use formations that are positionally stable in lift. For example, in the four ship "diamond" formation, the trail aircraft flies below the leader while the wing aircraft fly above the leader with their noses outboard of the lead's wingtip. This formation also results in stability of all of the moments for the trail aircraft and in pitch and side force for the wing aircraft. Aerial refueling is accomplished in a position stable in four of the five derivatives shown (all but side force).

For induced drag reduction, the preferred position is roughly 80-100% span outboard with no vertical separation. Below the vertical plane of symmetry, only one derivative is stable, above the plane, three are stable. This is one indicator of the difficulty of maintaining flight in this position.

CONCLUSION

A mathematical model for simulation of multiple aircraft in close formation flight has been proposed. The aircraft are assumed to be tailless. A slightly

modified form may be required for tailed aircraft. Six variables are included in the calculation of the aircraft-aircraft aerodynamic interactions. These are the three position variables, roll orientation, and the lift on each aircraft. To implement the model in a simulation, a single four dimensional table is required.

Six positional stability derivatives have been identified. Regions of stability have been mapped for the tailless study aircraft. Every region has at least one unstable mode. The region behind and below the lead aircraft (used for aerial refueling) is stable or neutral for five of the six modes. Regions for maximum drag reduction are unstable in two or four modes. These regions may change for tailed aircraft.

REFERENCES

1. Blake, W.B., and Multhopp, D., Design, Performance and Modeling Considerations for Close Formation Flight, *Proc. AIAA Atmospheric Flight Mechanics Conference*, (1998).

2. Gingras, D.R., Experimental Investigation of a Multi-Aircraft Formation, *AIAA Applied Aerodynamics Conference*, (1999).

3. McKeehen P.D., et al, Simulation of a Tailless Aircraft Using Non-Linear Indicial Response Aerodynamic Modeling Methods, *Proc. AIAA Modeling and Simulation Conference*, (1997).

4. McGowan, W.A., Calculated Normal Load Factors on Light Airplanes Traversing the Trailing Vortices of Heavy Transport Airplanes, NASA Technical Note D-829, (1961).

5. Bernstein, S., and Iversen, J.D., Dynamic Simulation of an Aircraft Under the Effect of Vortex Wake Turbulence, *Proc. AIAA Atmospheric Flight Mechanics Conference*, (1972).

6. Nelson, R.C., The Response of Aircraft Encountering Wake Turbulence, Air Force Flight Dynamics Laboratory Technical Report 74-29, (1974).

7. Sammonds, R.I., Stinnett, G.W. Jr, and Larson, W.E., Criteria Relating Wake Vortex Encounter Hazard to Aircraft Response, *Journal of Aircraft*, 14 (10), 981-987, (1977).

8. Hastings, E.C., Jr., and Keyser, G.L., Jr, Simulator Studies of Vortex Encounters by a Twin-Engine Commercial Jet Transport Airplane, NASA Technical Paper 1966, (1982).

9. Reimer, H.M., and Vicroy, D.D., A Preliminary Study of a Wake Vortex Encounter Hazard Boundary for a B737-100 Airplane, NASA Technical Memorandum 110223, (1996).

10. Weiss, J.M., Simulated Leadship Aerodynamic Interference as Applied in the B-2 Aircrew Training Device, *Proc AIAA Flight Simulation Conference*, (1991).

11. Hoak, D.E., et al, USAF Stability and Control Datcom, Air Force Wright Aeronautical Laboratory Technical Report 83-3084, (1978).

12. Albright, A.E., Dixon, C.J., and Hegedus, M.C., Modification and Validation of Conceptual Design Aerodynamic Prediction Method HASC95 with VTXCHN, NASA Contractor Report 4712, (1996).

13. Bloy, A.W., West, M.G., Lea, K.A., and Jouma'a, M., McGowan, W.A., Lateral Aerodynamic Interference Between Tanker and Received in Air-to-Air Refueling, *Journal of Aircraft*, 30 (5), 705-710, (1993).

APPENDIX – BENEFITS OF CLOSE FORMATION FLIGHT

Calculation of aircraft range or endurance under various constraints typically includes a term of the form C_L^k/C_D. With a parabolic drag polar, this can be written as follows:

$$\frac{C_L^k}{C_D} = \frac{C_L^k}{C_{Do} + C_L^2/\pi AR} \quad (1)$$

This quantity is maximized when:

$$C_{Do} = \frac{(2-k)}{k}\frac{C_L^2}{\pi AR} \quad (2)$$

The formation flight benefit (FFB) is defined as the increase in range or endurance of a formation of n identical aircraft relative to a single aircraft. Assuming that formation flight reduces the induced drag by a factor r, FFB can be written as:

$$FFB = \frac{C_{L,f}{}^{k} / (C_{Do} + rC_{L,f}{}^{2} / \pi AR)}{C_{L,s}{}^{k} / (C_{Do} + C_{L,s}{}^{2} / \pi AR)} \quad (3)$$

The lower bound on the induced drag reduction, r, is 1/n where n is the number of aircraft in the formation. This represents the case of aircraft that are joined. If we take the lift coefficient of the formation to be a constant λ times the lift coefficient of a single aircraft ($C_{L,f} = \lambda C_{L,s}$) and assume that the single aircraft operates at its maximum $C_L{}^k/C_D$ point, eq. 3 becomes:

$$FFB = \frac{2\lambda^{k}}{2 + k(r\lambda^{2} - 1)} \quad (4)$$

The lift ratio that maximizes FFB is easily shown to be $\sqrt{1/r}$. This indicates that a formation should fly at a higher lift coefficient than a single aircraft, which can be done by flying higher in altitude, slower in airspeed, or both. The unrestricted maximum FFB is:

$$FFB_{max} = r^{-k/2} \quad (5)$$

This indicates the benefit increases without limit with formation size. The benefit also increases with k, with the largest benefit for a formation of propeller powered aircraft on a long endurance mission (k=3/2). If we restrict the flight profile of the formation to that of a single aircraft (λ=1), which could arise from air traffic control restrictions, the benefit is sharply reduced, as shown in Table 1 for typical values of k used in performance analysis. For the k values given, the maximum benefit ranges from a 33% increase to a factor of four, much less than the infinite benefit of the unrestricted case.

k	FFB (λ=1)
1/2	(1+3r)/4r
2/3	(1+2r)/3r
1	(1+r)/2r
3/2	(3+r)/4r

Table 1. FFB for typical values of k with λ=1

If we assume the formation cruises at the same Mach number as a single aircraft, an analytical expression can be developed for the required increase in cruise altitude. We can write:

$$\lambda = \frac{C_{L,f}}{C_{L,s}} = \frac{2W/(\rho_f V_f{}^{2} S)}{2W/(\rho_s V_s{}^{2} S)} = \frac{\rho_s}{\rho_f} \quad (6)$$

The density variation in the Stratosphere in English units is:

$$\rho = 0.2971\rho_{oe}^{-\left(\frac{alt - 36089}{20806.7}\right)} \quad (7)$$

From this we can derive:

$$alt_f = alt_s + 20806.7 \ln \lambda \quad (8)$$

Variations in FFB with altitude increase are shown in figure 1 for various values of r. This figure corresponds to the case k=1, which maximizes lift to drag ratio. Vertical dashed lines indicate constant λ, which is the available lift constraint.

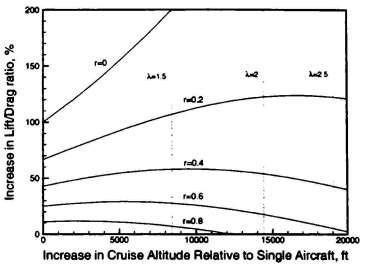


Figure A-1. Formation Flight Benefit

The key question is what are the attainable values of λ and r. λ represents the available lift constraint. While the upper bound of this is obviously wing stall, progressive flow separation over the wing results in a non-parabolic drag polar before stall. The Air Force Flight Test Center analytical performance model of the F-16A [A-1] includes a breakpoint where the drag polar becomes non-parabolic. Using this point to define the available lift, attainable values of λ for the F-16 decrease from 1.9 to 1.5 as Mach number increases from 0.4 to 0.9. This result is for a "clean" configuration of wingtip missiles only. Increases in C_{Do} increase the C_L for maximum L/D. A store loading of wingtip missiles, two external fuel tanks, six bombs and an ECM pod nearly doubles C_{Do}, reducing the attainable values for λ from 1.5 to 1.0.

The attainable reduction in induced drag, r, is more difficult to estimate. Beukenberg and Hummel [A-2] conducted two sets of flight tests with Do-28/Do-28 and Do-228/Do-28 lead/trail aircraft combinations. His second test included a flight controller which attempted to maintain the trail aircraft in a fixed position. He measured a 15% power reduction on the

trail aircraft over a 15 second interval and reduction. Using his published values for the lift coefficients and zero lift drag along with a parabolic drag polar, a value of r=0.84 is obtained for n=2. Schlichting [A-3] calculated the drag reduction of a formation representing each aircraft as a single horseshoe vortex. He derived an expression for the mutual induced drag between any two aircraft, using superposition to obtain the result for a formation. Assuming co-planar aircraft at the same lift coefficient, his result can be written using the present nomenclature as follows:

$$r = 1 + \frac{4}{\pi^2 n} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \ln\left[1 - \left(\frac{\pi b/4}{y_i - y_j}\right)^2\right] \qquad [9]$$

Lateral position is measured from the aircraft centerline, with Δy/b=1 corresponding to wing tips aligned. This expression is singular when the trailing vortex segments overlap (Δy/b=π/4). Maskew [A-4] conducted a series of vortex lattice calculations for three wings in a series of formations. He included an iterative trim procedure that maintained equal lift across the formation with zero rolling moment. His result for the maximum induced drag reduction (at Δy/b=0.9), is:

$$r = 0.2 + 0.8/n \qquad [10]$$

Hummel also conducted calculations with flat wakes (vortex lattice) and a rolled up wake model and found results similar to Maskew's. The various results are shown in Figure 2.

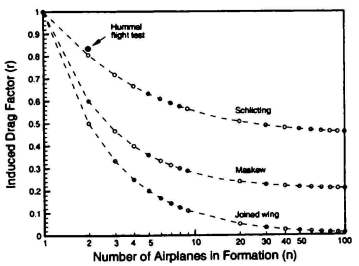


Figure 2. Variation in Induced Drag Reduction with Number of Airplanes

These results only consider basic aerodynamics only and neglect compressibility effects. Fuel consumption is also a significant factor in range and endurance. The performance of the engine at reduced power settings and increased altitude must also be considered.

A-1. McElroy, C.E., Performance Evaluation of the F-16A Airplane with F100-PW-100 Engine, Air Force Flight Test Center Technical Report 79-11 (1979).

A-2 Beukenberg, M. and Hummel, D., Aerodynamics, Performance and Control of Airplanes in Formation Flight, ICAS paper 90-5.9.3 (1990).

A-3 Hoerner, S.F., Fluid Dynamic Drag, published by the author (1965).

A-4. Maskew, B., Formation Flying Benefits Based on Vortex Lattice Calculations, NASA Contractor Report 151974 (1977).

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37262

AIAA-2000-4305

NEURAL-NETWORK-BASED MODELING OF ROTORCRAFT VIBRATION FOR REAL-TIME APPLICATIONS

Sesi Kottapalli
Aeromechanics Branch
Army/NASA Rotorcraft Division
NASA Ames Research Center
Moffett Field, California

Summary

The overall objective of this ongoing effort is to provide the capability to model and simulate rotorcraft aeromechanics behaviors in real-time. This would be accomplished by the addition of an aeromechanics element to an existing high-fidelity, real-time helicopter flight simulation. As a first step, the peak vertical vibration at the pilot floor location was considered in this neural-network-based study. The flight conditions considered were level flights, rolls, pushovers, pull-ups, autorotations, and landing flares. The NASA/Army UH-60A Airloads Program flight test database was the source of raw data. The present neural network training databases were created in a physically consistent manner. Two modeling approaches, with different physical assumptions, were considered. The first approach involved a "maneuver load factor" that was derived using the roll-angle and the pitch-rate. The second approach involved the three pilot control stick positions. The resulting, trained back-propagation neural networks were small, implying rapid execution. The present neural-network-based approach involving the peak pilot vibration was utilized in a quasi-static manner to simulate an extreme, time-varying pull-up maneuver. For the above pull-up maneuver, the maneuver load factor approach was better for real-time simulation, i.e., produced greater fidelity, as compared to the control stick positions approach. Thus, neural networks show promise for use in high-fidelity, real-time modeling of rotorcraft vibration.

Introduction

In order to expedite pilot training, it is important for any flight simulator to achieve a high degree of functional fidelity, i.e., the adequacy of piloted simulation. An example taken from currently used simulators is the Vertical Motion Simulator (VMS) at NASA Ames. The VMS is a high-fidelity, piloted, six degree-of-freedom, real-time flight simulator.[1-3] It allows for the greatest motion range of any flight simulator in the world. The Ames VMS can simulate a variety of aircraft including rotorcraft (not including some aeromechanics behaviors), the Space Shuttle Orbiter, and others. A vibration model of the UH-60A has been used for many years in the Ames VMS to simulate pilot seat-shake.[4] The associated flight test data based, seat-shaker algorithm does not involve neural networks.

Currently, rotorcraft aeromechanics behaviors are not adequately modeled in simulators. These aeromechanics behaviors include pilot vibration and cabin noise. Thus, future research could be directed towards inclusion of both cockpit vibration and noise. In this first-time study, helicopter vibration was modeled using neural networks. The non-inclusion of vibration in the simulations of severe and/or complex maneuvers could have an adverse effect on pilot performance because such maneuvers entail high vibration levels. It is thus important to extend the existing real-time simulation capabilities by the addition of rotorcraft aeromechanics behaviors, especially vibration.

Existing neural-network-based work covered real-time rotor system flight test load monitoring systems for the Navy SH-60 helicopter.[5-7] On-line evaluation of flight vibratory loads was studied.[5] The present study, however, covers neural-network-based, high-fidelity real-time ground based simulator modeling of pilot floor vertical vibration for the Army UH-60A helicopter. Neural network studies on rotorcraft performance, acoustics, and dynamics were initiated in the Army/NASA Rotorcraft Division at NASA Ames Research Center.[8-13]

The present neural-network-based results provide the capability to model rotorcraft aeromechanics behaviors in real-time. Specifically, the capabilities of the existing high-fidelity, real-time flight simulation would

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental Purposes. All other rights are reserved by the copyright owner.

be extended. This would be accomplished by the addition of an aeromechanics element.

For purposes of modeling the UH-60A vibration, two analytical approaches, involving different physical assumptions, were considered. This resulted in three neural-network-training databases. The first database involved a maneuver load factor that was derived using the helicopter roll-angle and its pitch-rate. The second and third databases involved the three pilot control stick positions. In the first and second databases, the helicopter gross weight was included as one of the neural network inputs. The third database was the same as the second database, except that the gross weight was not included.

In general, to obtain a time varying, step-by-step simulation of the pilot vibration during a maneuver, a neural network based time-series method can be used. However, such methods are complex. In the present, first-time modeling study using neural networks, a static-mapping approach involving the peak vibration level was followed. This implied that each flight condition was characterized by its peak vibration. The possibility of utilizing the present, peak-vibration-based static mapping in a quasi-static manner to simulate time varying maneuvers was also investigated. That is, the fidelity of a quasi-static, real-time simulation was studied. A quasi-static approach will not capture all dynamic effects, and may miss the prediction of relevant maximums and their associated phases. Also, a time-series analysis using neural networks will capture the maximums and phases more accurately, compared to a quasi-static approach. It should be noted that the present quasi-static approach represents one way of simulating maneuvers. The present work provides the capability of real-time simulation of pilot vibration for the entire UH-60A flight envelope.

The present use of neural networks was justified because of the following two reasons. First, trained neural networks can be rapidly executed, an advantage in real-time applications. Second, neural networks can perform multi-dimensional, nonlinear curve fitting, an advantage in high-fidelity applications. The present work is considered to be a generic methodology and is not specific to the presently considered rotorcraft configuration.

Objectives

The present neural-network-based modeling study involving the peak, N/rev pilot floor vertical vibration had the following three objectives:

1. Create two compact neural network training databases, one involving the helicopter body motions and the other involving the pilot controls.
2. Study the fidelity considerations using a) neural networks and the above maneuver load factor, and separately, b) neural networks and the three control stick positions.
3. Quantify the advantages and disadvantages of using the three training databases.

Vibration Neural Network Databases

The source of the presently used raw data was the NASA/Army UH-60A Airloads Program flight test database.[14, 15] The present study included the following flight conditions: level flights, rolls, pushovers, pull-ups, autorotations, and landing flares. The creation of the present three compact neural-network-training databases involved a substantial amount of manual effort and time.

Description of Present Databases

A single neural network output, common to all three training databases, was considered. The peak, N/rev pilot vertical floor vibration, peak PVV, was the neural network output. The overall approach used to obtain the peak PVV is discussed later, under "Database Construction Example I: Pull-up" and also under "Database Construction Example II: Autorotation."

Database 1 This neural network training database involved six inputs that are given as follows:

i) Advance ratio.
ii) Gross weight, lbs.
iii) Main rotor rotational speed, RPM.
iv) Density ratio.
v) Maneuver load factor, MLF, discussed below.
vi) Ascent/descent rate, fpm.

The MLF, a non-dimensional parameter, was used to characterize aircraft maneuvers involving simultaneous non-zero roll-angle and pitch-rate. In the present study, the MLF was defined by the following equation:

Maneuver load factor, MLF =
[1 / cos (roll-angle)] *
[1 + (pitch-rate * airspeed / g)]

(1)

where "g" is the acceleration due to gravity. The flight-path axis system[16] was used. The purpose of the MLF was to compactly represent complex maneuvers using a single, physics-based parameter. Depending on the reference axes system used, other parameters can be derived, and this would result in slightly different formulations.

Database 1 involved two aircraft parameters, the angle-of-bank and the pitch-rate. The other two databases do not involve these two parameters, but involve the three pilot control stick positions.

Database 2 This neural network training database involved eight inputs that are given as follows:

i) Advance ratio.
ii) Gross weight, lbs.
iii) Main rotor rotational speed, RPM.
iv) Density ratio.
v) Collective stick position, %
vi) Lateral stick position, %
vii) Longitudinal stick position, %
viii) Ascent/descent rate, fpm.

Database 3 This neural network training database was obtained after removing the gross weight from the database 2 input list. Thus, database 3 involved seven inputs that are given as follows:

i) Advance ratio.
ii) Main rotor rotational speed, RPM.
iii) Density ratio.
iv) Collective stick position, %
v) Lateral stick position, %
vi) Longitudinal stick position, %
vii) Ascent/descent rate, fpm.

The flight test data were represented in the neural network training databases in a physically consistent manner. Physically consistent refers to the manual extraction of the correct values of relevant parameters, e.g., the correct pitch-rate associated with a pull-up maneuver. Presently, the correct pitch-rate was taken as that corresponding to the peak PVV. Let the peak PVV occur at a time $t = \tau$. The correct pitch-rate was defined as that also occurring at time $t = \tau$. In general, the peak-PVV-time, τ was different for different maneuvers, and had to be individually determined.

Database Construction Example I: Pull-up

A pull-up maneuver at approximately 120 knots was considered in this example. In the UH-60A flight test database,[14,15] counter 11022 represented a pull-up maneuver. This maneuver represents an extreme maneuver. The pitch-rate associated with this particular maneuver is extreme, one that was employed intentionally for test purposes, or one that might be employed by the pilot to accomplish a sudden evasive action. For this flight condition, the gross weight was 16055 lbs, the rotor RPM was 255, and the advance ratio was 0.27. The time history record duration was 37 seconds (156 rotor revolutions).

The present study attempts to reconcile the aeromechanics and flight mechanics aspects. In this initial study, unfiltered time history records are shown. For flight mechanics considerations, helicopter body motions up to 10 read/sec are important.

Figure 1 shows the time history of the instantaneous pilot floor vertical acceleration for the above pull-up. To obtain a time varying, step-by-step simulation of the acceleration time history shown in Fig. 1, a neural network based time-series method can be used. In the present modeling study using neural networks, a static-mapping approach involving the peak vibration level was followed. Figure 2 shows the pitch-rate time history for the same counter, 11022. Figure 3 shows the corresponding time-histories of the three pilot control stick positions, the collective, longitudinal and lateral cyclics. It should be noted that the filtered pilot control stick position variations would not contain the high frequency components present in the unfiltered data shown in Fig. 3.

Pilot Floor Vertical Vibration Figure 4 shows the 4P component of the pilot floor vertical acceleration. This 4P component, Fig. 4, was obtained by breaking up the time history record, Fig. 1, into intervals of 8 revolutions each. In this study, the 4P component of the pilot floor vertical acceleration was referred to as the pilot vertical vibration, PVV. Presently, the PVV was obtained by performing a harmonic analysis[15] of the acceleration time-history in which the individual time-segments (time windows) were eight rotor revolutions long. Some dynamic effects may not be accurately modeled if the sample length used in the harmonic analysis is too large. At the same time, a too-small sample length will introduce spurious dynamic information. Future, detailed studies could focus on determining an appropriate sample length for the present application. Such a time window study would need to consider all maneuvers, make detailed comparisons of the resulting PVV's, and determine the appropriate sample length. Figure 4 shows that the peak PVV for the above pull-up was 0.22 g's.

Pitch-Rate The correct pitch-rate, used in calculating the maneuver load factor in database 1, for the above pull-up maneuver, counter 11022, was that corresponding to the peak PVV, Fig. 4. The correct pitch-rate, Fig. 2, was determined to be 9.6 deg/sec. The subject pull-up involved a negligible roll-angle, resulting in a maneuver load factor, MLF = 2.0.

Pilot Control Stick Positions The three control stick positions (databases 2 and 3) for the above pull-up maneuver, counter 11022, were manually obtained as those corresponding to the peak PVV, Fig. 4. The positions of the collective, lateral and longitudinal

cyclic controls at the time when the PVV was maximum were 76%, 60%, and 57%, respectively.

Database Construction Example II: Autorotation

An autorotation at approximately 60 knots was considered in this example. In the UH-60A flight test database,[14,15] the two counters 11539 and 11540 represented two segments of the selected single autorotation condition. For this flight condition, the approximate gross weight was 15910 lbs, the rotor RPM was 256, and the advance ratio was 0.17. The time history record duration for each segment was 27 seconds (115 rotor revolutions). The main rotor shaft power and the pilot collective stick position were the two helicopter performance parameters required to establish autorotation conditions.

Figure 5 shows the time histories of the pilot floor vertical acceleration and the main rotor shaft power for segment 1. The reduction in shaft power, starting just prior to 10 seconds, marks the beginning of the autorotation phase. Figure 6 shows the time history for the collective stick position for segment 1.

Figure 7 shows the time histories of the pilot floor vertical acceleration and the main rotor shaft power for segment 2. The resumption of power to the shaft occurs after 10 seconds and marks the end of the autorotation phase. Figure 8 shows the time history for the collective stick position for segment 2. Figures 7 and 8 show that the post-autorotation phase occurs in segment 2 and starts after 10 seconds.

Pilot Floor Vertical Vibration Figure 9 shows the 4P component of the pilot floor vertical acceleration for segments 1 and 2. The 4P component was obtained from the acceleration time histories shown in Figs. 5 and 7. Figure 9 shows that the peak 4P component of the pilot floor vertical acceleration for the complete autorotation condition occurred during its post-autorotation phase, segment 2 (for this particular maneuver). Figure 9 shows that the peak PVV was 0.26 g's.

For the above, complete autorotation condition, the neural network inputs needed in constructing the three databases were those corresponding to the peak PVV, Fig. 9, and involved data from segment 2, counter 11540.

Neural Network Approach

To accurately capture the required functional dependencies, the neural network inputs must be carefully selected and account for all important physical traits that are specific to the application. The important attributes of a neural network are its type (radial-basis function network or back-propagation network, etc.) and its complexity (i.e., the number of processing elements (PEs) and the number of hidden layers). The present overall neural network modeling approach[8-11] consists of first determining the best type of neural network to be used and then simplifying the network as much as is practical.

Determining the best type of neural network usually involves selecting either a radial-basis function (RBF) or a back-propagation network. The RBF network (Moody-Darken version) can be used in most situations in which one would consider using a back-propagation network.[17] In the present study, the back-propagation type of network was used.

Simplifying the network involves reducing the number of PEs and in a few cases, the number of hidden layers. The number of PEs required depends on the specific application. The determination of the appropriate number of PEs is done by starting with a minimum number of PEs. Additional PEs are added to improve neural network performance by reducing the RMS error between the test data and the neural network predictions. Typically, five PEs are added at each step in this process. Adding two or three PEs at a time fine-tunes the neural network.

If the correlation plot, comparing measured and predicted values, shows only small deviations from the 45-deg reference line, the neural network has produced an acceptable representation of the subject test data. If the plot shows points well off of the 45-deg line, the presence of "bad" test data is assumed. A detailed examination of the subject test database is then required to identify the source(s) of the errors associated with these test data.

The notation used in this paper to characterize a neural network is described as follows. A neural network architecture such as "4-25-5-1" refers to a neural network with four inputs, twenty five processing elements (PEs) in the first hidden layer, five PEs in the second hidden layer, and one output.

The application of neural networks to full-scale helicopter flight test vibration data was conducted using the neural networks package NeuralWorks Pro II/PLUS (version 5.2) by NeuralWare.[17] The present neural network RMS error was dimensionless and based on the squares of the errors for each processing element (PE) in the output layer. Generally, the RMS error was characterized by a monotonic decrease with the number of training iterations.[12] Also, any large differences in the magnitudes of the neural network variables were mitigated by appropriate scaling. In the present application, the cost function used in minimizing the

RMS error had equally weighted individual contributions.

The number of training data points was over 200. Approximately 25% of this training database involved maneuver-related points, in the maneuver categories referred to earlier. Here, maneuver-related refers to a flight condition for which the maneuver load factor, MLF $\neq$ 1. In the present study, the single neural network output was the UH-60A peak, pilot floor vertical vibration (peak PVV). Neural-network-based modeling results for the peak PVV are given in this paper in the form of correlation plots. The neural-network-predicted peak PVV is plotted versus the corresponding flight test peak PVV.

Results

Figure 10 shows the correlation obtained using the maneuver load factor, MLF, approach (database 1). Figure 10 shows the correlation plot from a MISO 6-10-5-1 back-propagation neural network. The back-propagation network was trained for 4 million iterations with a final RMS error of 0.07. Figure 10 shows that the corresponding error-band was +/- 0.05 g's. The maneuver load factor approach, database 1, gave acceptable results.

Figure 11 shows the correlation obtained using the control stick approach (database 2). Figure 11 shows the correlation plot from a MISO 8-10-5-1 back-propagation neural network. The back-propagation network was trained for 550,000 iterations with a final RMS error of 0.07. Figure 11 shows that the corresponding error-band was +/- 0.05 g's. The pilot control stick approach, database 2, gave acceptable results.

Figure 12 shows the correlation obtained using the control stick approach (database 3, gross weight not included). Figure 12 shows the correlation plot from a MISO 7-10-5-1 back-propagation neural network. The back-propagation network was trained for 1.75 million iterations with a final RMS error of 0.07. Figure 12 shows that the corresponding error-band was +/- 0.05 g's. The pilot control stick approach, database 3 with the helicopter gross weight not included, gave acceptable results.

To summarize the above results, Figs 10-12, both the maneuver load factor approach and the pilot control stick approach were shown to be reasonable approaches for statically-mapped vibration. In this context, Figs. 10-12 showed that the same level of modeling accuracy could be obtained from either approach. Also, the present, trained back-propagation neural networks were small, implying rapid execution.

A comparison of Figs. 10-12 brought up an interesting data-quality consideration. It should be noted in Figs 10-11, that a few experimental data points close to 0.10 g's were modeled to produce neural-network-based results that were close to the +0.05 g's error line. In Fig. 12, the neural-network-based result for one of the "0.10 g's" experimental point clearly fell outside of the +0.05 g's error line. An examination of database 3 showed that this point was associated with a roll-maneuver. This implied that it was necessary to include the gross weight as an input for maneuvers for the cases presently considered with the given approach.

Quasi-Static Real-Time Simulation

Selected results taken from Figs. 10-12 are shown in Table 1 in numerical form to show typical neural network predictions for real-time constant flight condition simulation. The test PVV's for four flight conditions and the neural-network-based PVV's, using all three databases, are shown in Table 1. The present neural-network-based model is good for high-speed level flight, descent, climb, and a constant turn flight condition, Table 1.

The present neural-network-based approach involving the peak vibration was utilized in a quasi-static manner to simulate an extreme, time-varying maneuver. This maneuver has been considered earlier, the pull-up at 120 knots, counter 11022. The pull-up maneuver's experimental time histories for the pilot floor vertical acceleration, the pitch-rate, the collective, the lateral and longitudinal cyclics, and the PVV (test PVV) were shown in Figs. 1-4. A quasi-static approach will not capture all dynamic effects, and may miss the prediction of relevant maximums and their associated phases. Also, a time-series analysis using neural networks will capture the maximums and phases more accurately, compared to a quasi-static approach.

In the present quasi-static approach, the time history variations shown in Figs. 2 and 3 were represented by their average values over an 8-revolution segment length. Approximately 20 averaged values were thus used. These averaged parameter values were used to prepare the neural network inputs for databases 1-3. The previously trained neural networks, Figs. 10-12, were subsequently executed using the above discrete-values-based input time histories. Thus, three quasi-static real-time simulations were obtained for the PVV.

Figures 13a and 13b show the present quasi-static, neural-network-based predictions for the extreme time-varying maneuver, the pull-up, using both the maneuver load factor approach and the control stick positions approach, respectively. The time segments for the test PVV, Fig. 4, and those used in preparing the neural network inputs in Figs. 13a and 13b, were

exactly aligned with each other in time, i.e., had a zero offset. Figure 13a shows that neural-network-based and the test peak PVV's were within 0.02 g's of each other. Figure 13a also shows that the neural-network-based and the test peak PVV phases differed by approximately 1 second. The control stick positions approach, Fig. 13b, resulted in an overshoot of the PVV amplitude. The overshoot ranged from approximately 0.04 g's (database 2) to 0.10 g's (database 3). The peak PVV phase predictions differed by approximately 3 seconds for both cases, databases 2 and 3.

Figures 14a and 14b show exactly the same neural network results as in Figs. 13a and 13b, except that the Fig. 14 neural network results were offset by one half-segment. This was done to maintain physical consistency, i.e., in the time domain the neural network inputs must precede the resulting vibration that is being modeled, Figs. 14a and 14b. An appropriate value of the above offset can be determined by detailed parametric studies and was not done here. Figure 14a shows that neural-network-based and the test peak PVV's were within 0.02 g's of each other. Figure 14a also shows that the neural-network-based and the test peak PVV phases were approximately coincident.

For the pull-up maneuver under consideration, the maneuver load factor approach, Fig. 14a, gave better real-time simulation, i.e., resulted in greater overall fidelity with reasonably accurate peak prediction and only slight phasing differences, as compared to the control stick positions approach, Fig. 14b. The control stick positions approach again resulted in an overshoot of the peak PVV amplitude, Fig. 14b. The overshoot ranged from approximately 0.04 g's (database 2) to 0.10 g's (database 3). The peak PVV phase predictions differed by approximately 2 seconds for both cases, databases 2 and 3. The present use of a one half-segment offset was strictly empirical, and worked for the presently considered maneuver. A thorough and systematic variation of the offset involving the optimization of an appropriate cost function is required to obtain the best value of this important offset.

There are noticeable phase differences between the test PVV maximum and the neural-network-based PVV maximums for the control stick cases, Figs. 13b and 14b. This implies that in future studies, the time lag between the pilot control stick position inputs (collective, lateral and longitudinal cyclics) and the resulting PVV must also be accounted for when preparing the neural network inputs.

Conclusions

The present neural-network-based results showed that it was feasible to obtain reasonably accurate, real-time models of rotorcraft vibration under various flight conditions. The flight conditions considered were as follows: level flights, rolls, pushovers, pull-ups, autorotations, and landing flares.

For purposes of modeling UH-60A vibration, two analytical approaches were considered. This resulted in three neural network training databases. The first database involved a maneuver load factor that was derived using the helicopter roll-angle and its pitch-rate. The second and third databases involved the three pilot control stick positions. In the first and second databases, the helicopter gross weight was included as one of the neural network inputs. The third database was the same as the second database, except that the gross weight was not included. The resulting, trained back-propagation neural networks were small, implying rapid execution.

The present neural-network-based approach involving the peak pilot vibration was utilized in a quasi-static manner to simulate an extreme, time-varying pull-up maneuver. For the above pull-up maneuver, the maneuver load factor approach was better for real-time simulation, i.e., produced greater fidelity, as compared to the control stick positions approach. Thus, neural networks show promise for use in high-fidelity, real-time modeling of rotorcraft vibration for piloted simulations.

Acknowledgements

The author wishes to thank Bill Hindson, Jay Shively, Matt Whalley, and Bob Chen, all of NASA Ames Research Center, for their feedback and constructive suggestions.

References

1. Aiken, E.W., Lebacqz, J.V., Chen, R.T.N. and Key, D.L., "Rotorcraft Handling-Qualities Design Criteria Development," NASA Conference Publication 2495, Volume II, NASA/Army Rotorcraft Technology Conference, March 1987.

2. Chen, R.T.N, Lebacqz, J.V., Aiken, E.W., and Tischler, M.B., "Helicopter Mathematical Models and Control Law Development for Handling Qualities Research," NASA Conference Publication 2495, Volume II, NASA/Army Rotorcraft Technology Conference, March 1987.

3. Sweeney, C., Sheppard, S, and Chetelat, M., "Development and Operation of a Real-Time Simulation at the NASA Ames Vertical Motion Simulator," AIAA Flight Simulation Technologies Conference, AIAA-93-3575, Monterey, California, August 1993.

4. Plonsky, J.G., Development of Equations to Improve Deficiencies in the GenHel UH-60A Math Model, Sikorsky Aircraft Division, United Technologies Corporation, NAS2-11058, 1989.

5. McCool, K.M., Flitter, L.A., aand Haas, D.J., "Development and Flight Test Evaluation of a Rotor System Load Monitoring Technology," American Helicopter Society 54th Annual Forum, Virginia Beach, Virginia, May 1998.

6. Haas, D.J., Milano, J., Flitter, L., "Prediction of Helicopter Component Loads Using Neural Networks," *Journal of the American Helicopter Society*, January 1995, pp.72-82.

7. McCool, K.M. and Haas, D.J., "Prediction of Helicopter Airspeed and Sideslip Angle in the Low Speed Environment," McCool K.M. and Haas, D.J., American Helicopter Society 53rd Annual Forum, Virginia Beach, Virginia, April-May 1997.

8. Kottapalli, S. "Neural Network Research on Validating Experimental Tilt-Rotor Performance," to be published in the July 2000 issue of the *Journal of the American Helicopter Society.* AIAA 16th Applied Aerodynamics Conference, AIAA-98-2418, Albuquerque, New Mexico, June 1998.

9. Kottapalli, S. "Application of Neural Networks to Aeromechanics Problems," 24th European Rotorcraft Forum, Marseilles, France, September 1998.

10. Kottapalli, S., "Exploratory Study on Neural Control of Rotor Noise and Hub Loads," American Helicopter Society Technical Specialists' Meeting for Rotorcraft Acoustics and Aerodynamics, Williamsburg, Virginia, October 1997.

11. Kottapalli, S., "Identification and Control of Rotorcraft Hub Loads Using Neural Networks," American Helicopter Society 53rd Annual Forum, Virginia Beach, Virginia, April-May 1997.

12. Kottapalli, S., Abrego, A., and Jacklin, S., "Application of Neural Networks to Model and Predict Rotorcraft Hub Loads," American Helicopter Society Second International Aeromechanics Specialists Conference, Bridgeport, Connecticut, October 1995.

13. Kottapalli, S. and Kitaplioglu, C., "Neural Network Representation of Experimental Tilt-Rotor Noise," 6th AIAA/CEAS Aeroacoustics Conference, AIAA-2000-1924, Maui, Hawaii, June 2000.

14. Kufeld, R.M., Balough, D.L., Cross, J.L., Studebaker, K.F., Jennison, C.D., and Bousman, W.G., "Flight Testing of the UH-60A Airloads Aircraft," American Helicopter Society 50th Annual Forum, Washington, D.C., May 1994.

15. Bondi, M.J. and Bjorkman, W.S., "TRENDS, A Flight Test Relational Database," User's Guide and Reference Manual, NASA TM 108806, June 1994.

16. Chen, R.T.N. and Jeske, J.A., "Kinematic Properties of the Helicopter in Coordinated Turns," NASA Technical Paper 1773, April 1981.

17. NeuralWorks Manuals: a) Reference Guide b) Neural Computing c) Using NeuralWorks, NeuralWare, Inc., Pittsburgh, Pennsylvania, 1995.

Table 1. Neural-Network-Based Predictions of Pilot Floor Vertical Vibration, PVV, g's

Flight Condition	Test	Maneuver Load Factor Database 1	Control Stick Database 2	Control Stick, No Weight Database 3
Level flight, 135 knots	0.10	0.09	0.10	0.11
Descent, 160 knots	0.25	0.24	0.25	0.26
Climb, 62 knots	0.12	0.12	0.12	0.12
Turn, 45 deg, 145 knots	0.13	0.13	0.13	0.14

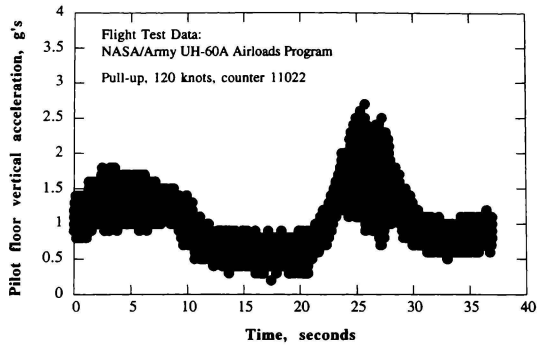


Fig. 1. Time history of pilot floor vertical acceleration, pull-up.

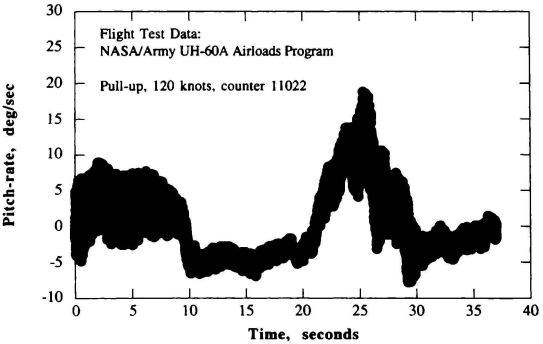


Fig. 2. Time-history of aircraft pitch-rate, pull-up.

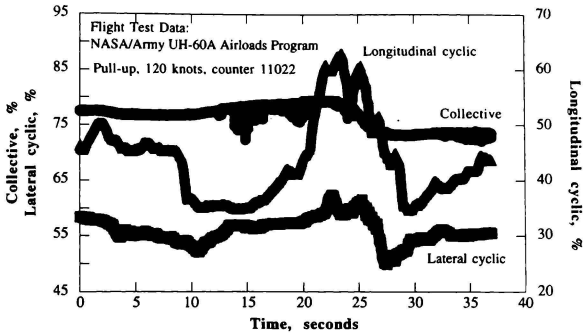


Fig. 3. Time histories of pilot control stick positions, pull-up.

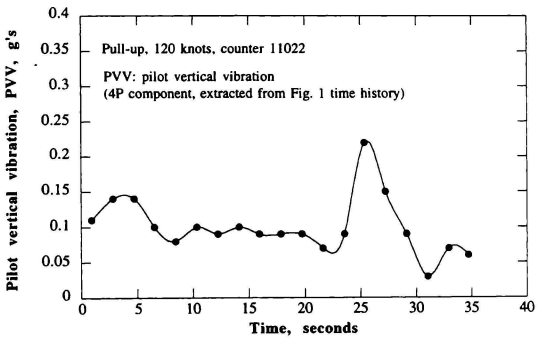


Fig. 4. Pilot vertical vibration, pull-up.

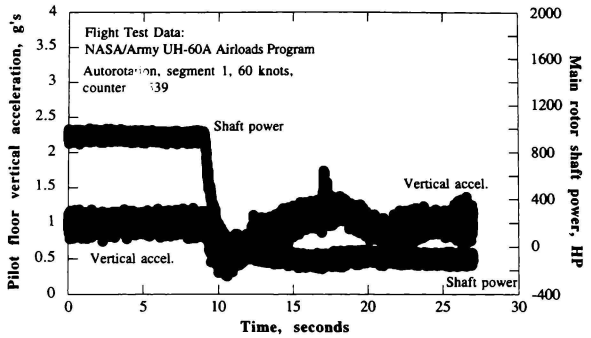


Fig. 5. Time histories of pilot floor vertical acceleration and rotor shaft power, autorotation, segment 1.

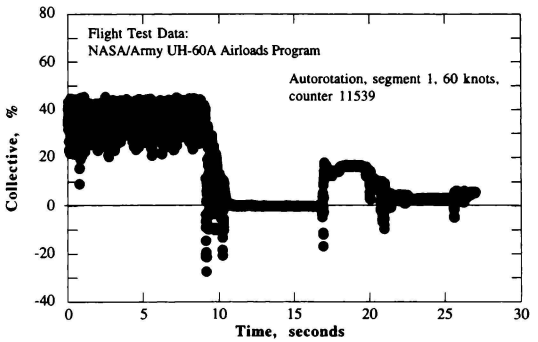


Fig. 6. Time history of collective stick position, autorotation, segment 1.

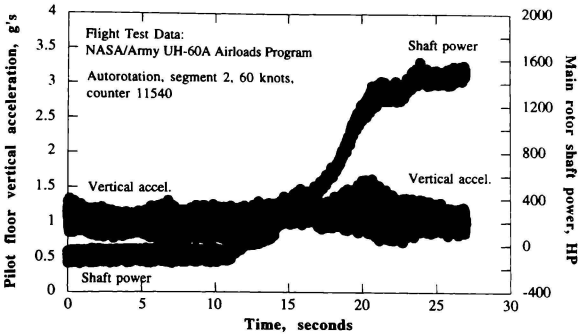


Fig. 7. Time histories of pilot floor vertical acceleration and rotor shaft power, autorotation, segment 2.

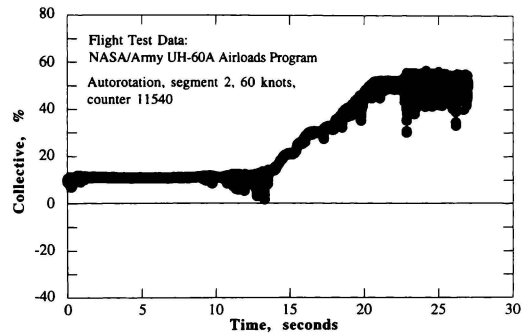


Fig. 8. Time history of collective stick position, autorotation, segment 2.

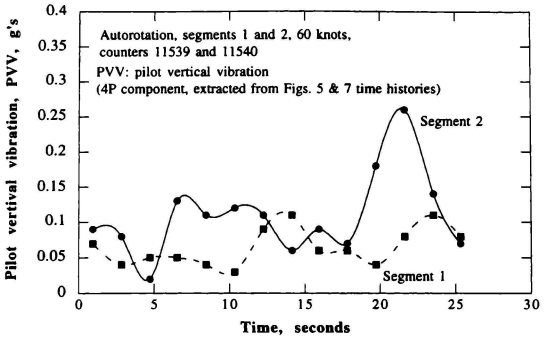


Fig. 9. Pilot vertical vibration, autorotation, segments 1 and 2.

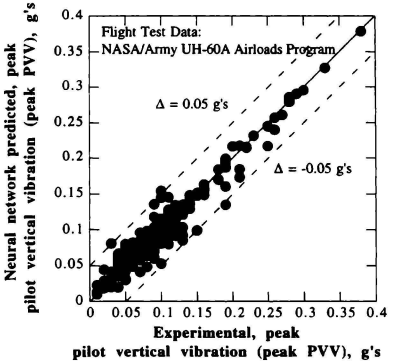


Fig. 10. Correlation, using "maneuver load factor," database 1.

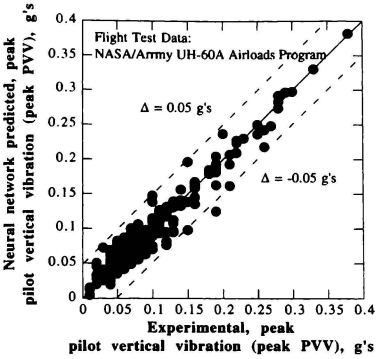


Fig. 11. Correlation, using pilot control stick positions, database 2.

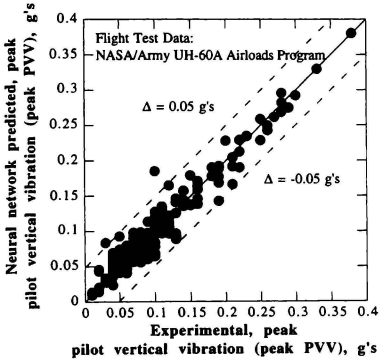


Fig. 12. Correlation, using pilot control stick positions, weight not included, database 3.

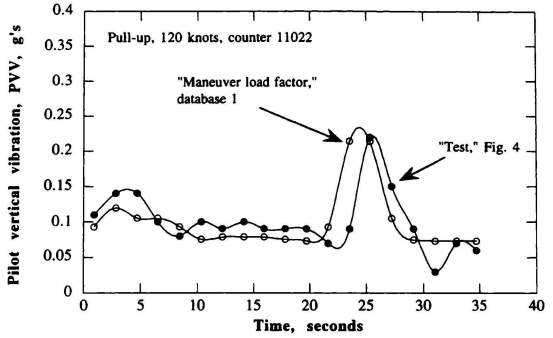


Fig. 13a. Quasi-static real-time simulation using maneuver load factor, "zero offset," database 1.

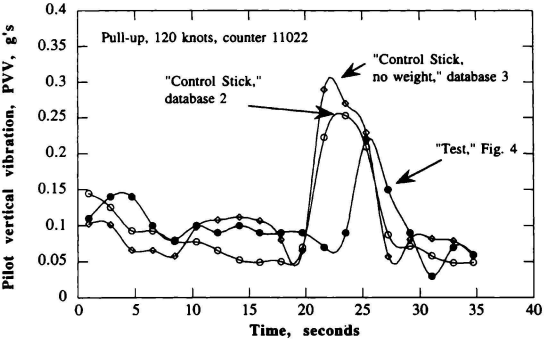


Fig. 13b. Quasi-static real-time simulation using control stick positions, "zero offset," databases 2-3.

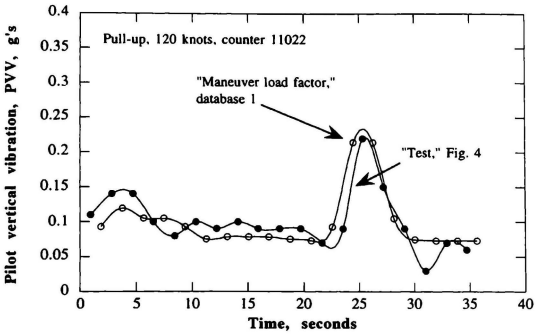


Fig. 14a. Quasi-static real-time simulation using maneuver load factor, "one half-segment offset," database 1.

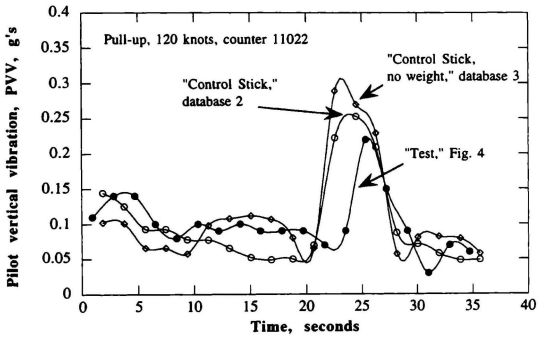


Fig. 14b. Quasi-static real-time simulation using control stick positions, "one half-segment offset," databases 2-3.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37299

AIAA-2000-4307

AN ADVANCED ON-BOARD AIRDROP PLANNER TO FACILITATE PRECISION PAYLOAD DELIVERY

Philip D. Hattis, AIAA Associate Fellow
Laboratory Technical Staff, C. S. Draper Laboratory - Cambridge, Massachusetts
Thomas J. Fill
Principal Member of the Technical Staff, C. S. Draper Laboratory - Cambridge, Massachusetts
David S. Rubenstein, AIAA Senior Member
Senior Member of the Technical Staff, C. S. Draper Laboratory - Cambridge, Massachusetts
Robert P. Wright
Assistant Vice President, Mid-Atlantic Group, Planning Systems, Inc. - McLean, Virginia
Richard J. Benney, AIAA Senior Member
Team Leader Airdrop Technology Team, US Army Soldier and Biological Chemical Command - Natick, Massachusetts

ABSTRACT

Precision airdrop from high altitude is increasingly important for delivery of military and humanitarian supplies. A personal computer (PC)-based, C-language, advanced airdrop planner is being developed as part of the New World Vistas Precision Aerial Delivery program in parallel with an advanced PC-based wind modeling tool. The combined operation of these two new tools on-board airdrop carrier aircraft will enable more accurate delivery of high altitude airdrop payloads by accounting for actual flight circumstances, by utilizing forecast and in-flight environmental measurements, and by providing timely, high value airdrop advisory information to the carrier aircraft crew. Embedded in the planner is a six-degree-of-freedom simulation of ballistic parachutes that has already been developed and tested. The planner software architecture as well as its user and carrier aircraft interfaces have been defined, with implementation planned by 2001. The planner and wind modeling tool will be demonstrated together on-board a C-130 aircraft in 2001. The features and design of the planner are discussed, some example ballistic airdrop simulation results are presented, and the operational procedures for using the planner in combination with the wind modeling tool are reviewed.

INTRODUCTION

The New World Vistas Precision Aerial Delivery (NWV-PAD) research initiative is a four year basic research program sponsored by the Air Force Office of Scientific Research and managed by the US Army Natick Soldier Center (Natick) in partnership with a number of Air Force organizations. Both Natick and the Army Yuma Proving Ground (YPG) have contributed significantly to the program. The program objective is to explore technologies that will substantially improve the delivery accuracy of 1000-2200 pound payloads via high altitude airdrop. The program also seeks to minimize the cost per pound of payload while maintaining a tight ground delivery accuracy. Synergistic Air Force/Army advancements in airdrop systems, real-time onboard airdrop-related weather data, and computed air release point (CARP) mission planning capabilities are all being explored. The ultimate goal is to rapidly transition technologies that will provide the user community with a "just-in-time point-of-use precision aerial delivery capability".

The technologies being pursued include advanced ballistic and controllable parachute systems. The Affordable Guided Airdrop System (AGAS)[1] is a candidate airdrop system that utilizes a standard G-12 resupply cargo parachute with an A-22 container payload (2200 lb capacity). The G-12 is modified for AGAS by extending the risers with four pneumatic muscle actuators (PMAs)[2]. PMAs are thin braided fabric tubes that contract in length with great force when pressurized. The AGAS is controlled by pulling riser slips with combinations of the PMAs. In addition, the integration of an advanced, near-real-time CARP mission planning algorithm with the aircraft and the payload(s) is being investigated and is reported in this paper. These combined capabilities have the potential to improve the accuracy of all Department of Defense airdrop systems at all altitudes. Quantifying these improvements is part of the program.

Weather sensing and prediction for NWV-PAD-based airdrops will apply the most current wind and density field predictions from the Mesoscale Model, 5th-Generation (MM5) that is run by the Air Force Weather

Copyright © 2000 by C. S. Draper Laboratory, Inc. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

Agency (AFWA). Airborne demonstrations at the YPG will apply high-resolution MM5 predictions from the Army 4DWX program that simulates AFWA support. This data will be used within an onboard software package that is also discussed in this paper that fuses the weather forecast with additional wind and density information collected en-route and over the drop zone (DZ).

Additional complementary technologies are also being investigated including modeling of airdrop system fluid-structure interactions. This work is coupling computational fluid dynamics and structural dynamics software to predict airdrop system performance[3].

The NWV-PAD program will integrate these efforts into a flight demonstration in 2001. Precision airdrop can increase the survivability of Air Force transport aircraft for all airdrop operations by enabling increased drop altitudes and increased horizontal stand-off in strong wind fields. Precision airdrop also provides new logistics capabilities to the commander in the field. The NWV-PAD technologies can be made compatible with C-130, C-141, C-17 and C-5 aircraft. The demonstration will utilize a C-130. The requirements for the Army Vision for Beyond 2010 (AB2010) and the future Objective Force highlight the need for aerial delivery resupply in future conflicts. In addition, other-than-conventional-war scenarios have shown the need for increased accuracy for humanitarian airdrops that would generally be done from high altitude. These future operational requirements demand precise high altitude airdrop capability. The NWV-PAD technologies will dramatically improve the survivability of aircraft during airdrop missions for AB2010, and will do so affordably.

In 1998 the C.S. Draper Laboratory (CSDL) began work under the NWV-PAD program to develop an advanced airdrop simulation and carrier aircraft onboard precision airdrop planning capability. Planning Systems, Inc. (PSI) began NWV-PAD program work at about the same time to develop a carrier aircraft onboard computational tool to model high fidelity wind fields near the airdrop DZ. The PSI tool, called WindPADS (Wind-profile Precision Air Delivery Systems), will provide data to enable the CSDL airdrop planner to output advisory information to a carrier aircraft crew needed to provide in-flight operational support of airdrop operations and to improve the accuracy of airdrop payload delivery. The CSDL onboard planner operates the airdrop simulation on a personal computer (PC) with new user interfaces to facilitate improved-accuracy ballistic parachute drop procedures. The planner also supports actively guided airdrop systems by outputting reference decent trajectories for guidance algorithm use.

The CSDL NWV-PAD program goals include: Enabling use of an applicable airdrop simulation to support precise CARP determination; facilitating the simulation's use within a PC-based airdrop mission planning facility onboard a carrier aircraft; providing the hooks for the simulator to support guidance, navigation, and control system (GN&C) design and evaluation for low lift-to-drag ratio (L/D) airdrop systems that may use a variety of navigation sensor combinations. An important part of the CSDL airdrop planner capability is its ability to utilize the PSI wind models.

The PSI NWV-PAD program objective is to design and demonstrate a self-contained, onboard communications and data processing system (WindPADS) to produce high-resolution, high-fidelity models of the atmosphere over the DZ to support CARP determination by the CSDL airdrop planner in all weather over all types of terrain. The supporting PSI program goals include: Obtaining in-situ atmospheric measurements (wind/density) near the DZ using dropsondes ejected from the carrier aircraft; assimilating in-situ atmospheric measurements (dropsonde and flight-level data) with high-resolution mesoscale atmospheric dynamic prediction models; establishing internal and external data interfaces; system optimization through trade-off engineering studies.

Work on the advanced airdrop planner and supporting simulation began in 1998 with the development of specifications for the simulation and planning tool. In 1999 work began on the planner and simulation design and development against the specifications. This work included development of a new simulation around a core capability from a parafoil airdrop simulation developed under the previous Precision Guided Airdrop System (PGAS) program. Work also proceeded in 1999 to develop the high-level design of a PC-based airdrop planning tool that can be flown and operated on-board carrier aircraft for demonstration. Work on the planner and simulation in 2000 is focussed on completing their PC-based implementation. A flight demonstration is expected in 2001.

Work on the WindPADS concept-of-operations began in mid-1998 with application of the MM5 mesoscale model, at various horizontal grid resolutions (9 km, 3 km, 1 km), using atmospheric data from actual YPG parachute drops. Observed trajectories were compared with those from MM5 predictions at various resolutions, as well as those determined from ground-based wind measurements using an available, less-sophisticated descent trajectory algorithm. In 1999, design of the atmospheric three dimensional (3D) field interface between WindPADS and the CSDL airdrop planner was completed. This standardized interface enables the transfer of wind field, and other

atmospheric data, from the WindPADS PC to the PC with the on-board airdrop planner to support simulations and engineering studies using the NWV-PAD simulator.

Operational MM5 12 km fields were obtained for the area surrounding Grand Junction, Colorado to test the CSDL airdrop planner interface and enable sensitivity studies and simulations for a region of rugged/complex terrain. Components of the National Oceanographic and Atmospheric Administration Forecast Systems Laboratory Local Analysis and Prediction System (LAPS) were obtained, implemented and tested for assimilating in-situ atmospheric measurements with MM5 predicted fields. The LAPS components produce atmospheric flows from MM5 predicted "background" fields consistent with underlying topography, atmospheric stability, and in-situ measurements. MM5 high-resolution (1.33 km) forecast fields for Grand Junction were provided to CSDL to test the interface and support NWV-PAD airdrop simulations and preparation for airborne demonstrations. Still planned WindPADS work includes construction and testing with the intended onboard PC, testing dropsonde data acquisition (laboratory and airborne), developing and testing dropsonde data assimilation modes (flat and complex terrain), testing interfaces with the CSDL airdrop planner (ground and airborne), and planning and participating in airborne demonstrations including airdrops at YPG in 2001.

The following topics are addressed in this paper: The features of the NWV-PAD airdrop simulation and wrap-around planner, with reviews of their architecture, and sample simulation results; details of the features of the new high-fidelity wind models and the basis of the wind models; an overview of the combined on-board operational strategy of the planner and wind model data generation system; a summary of the program status and conclusions.

NWV-PAD AIRDROP SIMULATION AND PLANNER

The Simulation Goals, Features, and Architecture

Additions and modifications to the PGAS simulation for the NWV-PAD program have enabled its use for ballistic and low L/D airdrop simulations. The PGAS simulation was developed under a contract awarded to CSDL in 1994. The program developed flight code for the autonomous GN&C of ram-air parafoil systems. To support development of the autonomous airdrop system flight code and evaluation of the demonstration vehicle flight tests, the PGAS program also developed a six-degree-of-freedom (6 DOF) simulation representing the parafoil and payload system dynamics. During the PGAS program, models of small and large parafoils were implemented in the simulation.

Key objectives of the NWV-PAD program for the simulation have been:

- Incorporation of ballistic airdrop system models into the core PGAS simulation.
- Conversion of the simulation from Matlab to the C language.
- Enabling use of high fidelity wind models.
- Addition of a Monte-Carlo analysis capability.
- Incorporation of user-friendly interfaces.

These changes support the following NWV-PAD application goals:

- Application to ballistic parachutes.
- Much reduced processing time.
- Portability across a variety of PC-class computational platforms.
- Calculation of the expected payload landing dispersion footprints.
- Generation of reference descent trajectory data for use by guided airdrop systems.

A discussion of the PGAS core simulation design and capabilities is available in a previous AIAA paper[4].

While many of the NWV-PAD airdrop simulation models were derived from the PGAS simulation, the PGAS simulation was originally restricted to address flight of a fully deployed ram-air parafoil. The NWV-PAD objectives require that the airdrop simulation be capable of modeling a variety of airdrop systems, each with its own unique set of flight phases, from a point prior to release inside the aircraft to touchdown. These airdrop systems fall into three categories: High-altitude-high-opening systems; high-altitude-low-opening (HALO) systems; low-altitude-low-opening systems. To support all these systems, as well as provide for modeling of the pre-release captive-carry flight and post-release flight segments, the NWV-PAD simulation was implemented with seven unique flight phases:

• Captive	Captive-carry flight of the airdrop system by the carrier aircraft
• Extract	Release of the airdrop system to exit from the carrier aircraft
• Stabilize	Initial stabilization after exit (airdrop system dependent)
• High-v-Descent	High speed descent for HALO airdrop systems
• High-v-Deploy	Main canopy deployment for HALO airdrop systems
• Equilibrium Glide	Main canopy fully deployed descent model
• Landing	After ground impact

Full flexibility is provided within the simulation to specify which phases are treated and what models are used in each phase.

A preliminary set of models has been implemented in the simulation for each of the phases to support airdrop planner development and testing of CARP predictions for ballistic systems released from a C-130 carrier. The simulation structure can support use of either analytical or table look-up-based models for any of the seven flight phases. New models can also be incorporated with little difficulty.

Figure 1 illustrates the layout of the execution modes that are provided in the NWV-PAD simulation. When the simulation is used as a planner element, following completion of the simulation and Monte-Carlo loops, the desired payload landing location and expected wind conditions are fed back to the planning tool for update of the desired release location and approach flight conditions. These release updates are then used to re-initialize the simulation subroutine on subsequent calls until the desired landing target is acquired. A serial input/output interface is identified in Figure 1 for transfer of necessary information to guided airdrop systems. The simulation provides interfaces with the CARP routines, and desired future capabilities that include a zoomable map display interface and a "pilot" graphical user interface (GUI). The simulation accounts for flight en-route to the desired release point and accommodates models of airdrop system release and glide-to-touchdown for both guided and ballistic airdrop systems.

The simulation has been structured to use atmospheric and wind data from high fidelity wind models discussed later in this paper. However, the simulation also has a set of self-contained standard atmospheric and wind models for use in lieu of the high fidelity model.

The Simulation's Ballistic Parachute Model

The major enhancement of the PGAS simulation for the NWV-PAD application was addition of a ballistic parachute descent model that enabled round parachutes to be simulated. The existing PGAS parafoil model served as the starting point for the design. The Matlab version of this simulation was first modified to match flight test data received from Natick. The resulting model was then coded in the C language and incorporated into the NWV-PAD simulation. Design emphasis so far was given to ballistic parachute descent phases (the high-v-descent and equilibrium-glide phases) with a 6 DOF model capability. Variations of the ballistic airdrop model in the simulation support treatment of a steady ballistic descent using a G-12D parachute canopy model or a cross parachute model. These models were developed and tuned to the appropriate flight test data.

Because of limited test data for the ballistic parachute models, forces on the parachute and payload were estimated, and resulting moments about the center-of-gravity were computed using an assumed location of the aerodynamic center. The available ballistic parachute dynamics data was deemed insufficient to include a measured moment coefficient model. Aerodynamic damping terms are in the new model, but in the current absence of adequate flight data the values are set to zero.

Some Sample Simulation Results

Ballistic parachute trajectories that apply a model of the Army G-12 parachute have been generated by the NWV-PAD simulation and validated against flight test data. In the example provided here, the simulation depicts descent data for a 64 ft diameter G-12D canopy, weighing 125 pounds. The attached payload is a 4 ft cube weighing 2,200 lb. The mass properties of the parachute were computed assuming the properties of a hemisphere. The initial altitude is roughly 14,700 ft. The initial cross-range from the target is 6,850 ft. (Note that "target" here refers to the location of touchdown of the associated flight-test data from an actual, instrumented airdrop.) Wind velocities obtained from the G-12 test drop flight data were used in the simulation. Figures 2 and 3 co-plot the simulation and flight test trajectory data. Figure 2 shows the altitude vs. time. Figure 3 shows the cross-range vs. time, with significant plot curvature associated with wind effects. The wind starts out predominantly easterly at release of the airdrop system from the carrier aircraft. At an altitude of roughly 5,000 ft, the easterly component of the wind rapidly diminishes to near zero and remains small in that direction for the remainder of the descent. Below 5,000 ft, the airdrop system has a small, mostly northerly wind-induced drift. The trajectory cross range for the simulation and flight test in Figure 3 vary by less than 200 ft until approximately 300 seconds. The major factor in the mismatch after that time is nonzero lateral forces that are not included in the current simulation models but are actually present. This effect is accentuated at lower altitudes. The final cross-range disparity between the simulation and the flight data is roughly 500 ft. Figure 2 shows the maximum difference between the simulation altitude and the actual flight-test altitude is approximately 500 ft, which is attributable to current approximations in the model of the G-12 parachute drag coefficient. Additional flight test data to be collected on instrumented G-12 parachute airdrops will be used to improve the simulation models, thereby reducing the discrepancies between simulation-predicted trajectories and flight-test trajectories.

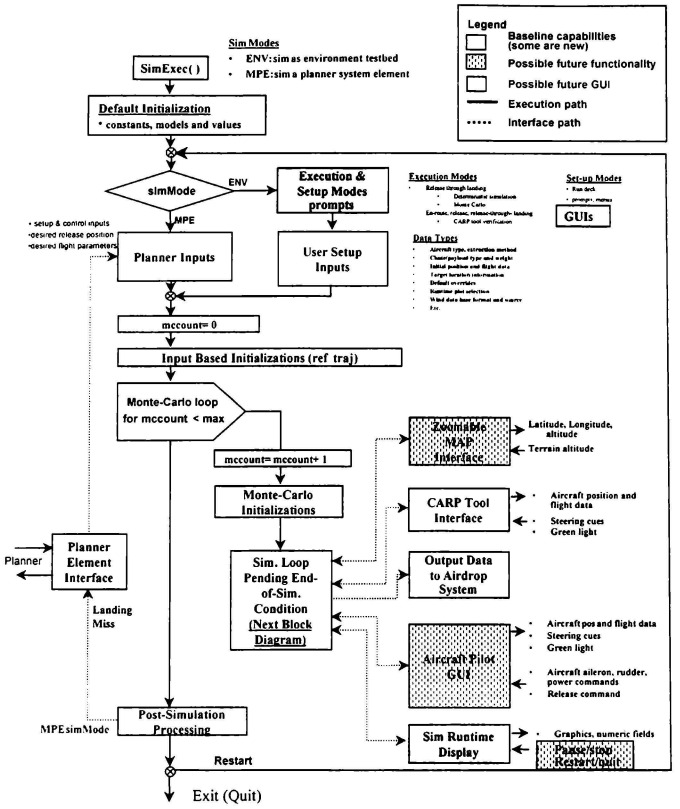


Figure 1. Overall NWV-PAD Airdrop Simulation Block Diagram

The Planner

The PC-based airdrop delivery planning tool will be demonstrated on-board a carrier aircraft in 2001. It will calculate the ballistic parachute/payload trajectories based on models of the airdrop system being released, near-real-time wind profile models, and carrier aircraft state information. The NWV-PAD program goals for the planner are CARP advancement and improvement by integration of planning, simulation, 3D/4D wind predictions, and carrier aircraft data en-route to the airdrop payload DZ. The airdrop planner would then be able to update and refine the projected payload release point during the carrier aircraft flight. The tool will also enable addition of an output interface to communicate reference trajectory data needed by guided airdrop loads.

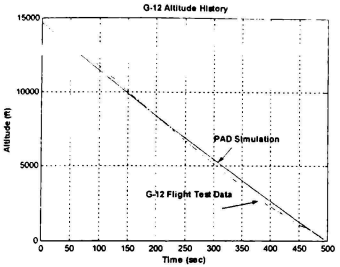


Figure 2. Simulation Comparison to Flight Test Data - Altitude vs. Time

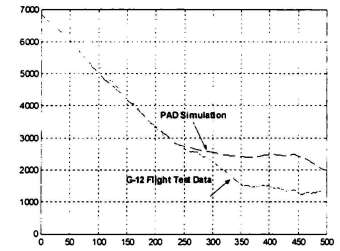


Figure 3. Simulation Comparison to Flight Test Data - Cross Range (ft) vs. Time (sec)

The functional layout of the planning tool elements and its external interfaces are shown in Figure 4. The GUI provides the user with executive control of the planning tool. Following initialization, the GUI algorithm enters an event loop in which it waits for operator inputs. Access (i.e., callbacks) to the planning tool elements are determined by the processing requirements and the user interfaces.

Most user inputs to the planner will prompt the execution of the range/validity check element to protect against input of erroneous data or a load attempt with incomplete data. Input of the necessary data initiates the planner processing.

Planning tool processing is divided into two flight phases or modes: The setup (pre-flight) mode; the en-route (flight) mode. The setup mode is where airdrop characteristics and CARP input sets are defined, entered, and evaluated for initial CARP estimates by the planner. The en-route mode operates while the aircraft flies to the DZ with the planning tool providing status information relative to the achievement of both the CARP and desired release conditions. During en-route mode operations, the planner utilizes carrier aircraft state information to derive CARP-relative time, position, velocity, and attitude metrics which the flight crew can apply to achieve an accurate and timely payload release.

The planner's CARP solution has a deterministic part and a statistical part. The CARP process loop has responsibility for computing and maintaining both. The deterministic solution part must be converged and updated in response to any new data and the solution status is provided to the user. This solution part provides the nominal release point to deliver the initial airdrop load to the desired impact point. The CARP process is designed to provide a three-tiered convergence status based on *a priori* landing miss distance thresholds. A "void" status is indicated if no solution is available or the error is excessive. An "un-converged" status is indicated if the impact point miss (IPM) is outside a specified tolerance but not excessive. Finally, a "converged" status is indicated when the IPM is within the specified tolerance. The statistical solution provides the expected envelope of the airdrop system delivery given variations in the environment, vehicle dynamics, and human-derived effects. The planning tool utilizes Monte-Carlo analysis methods and simulation to derive the statistical solution. The CARP process loop must accumulate the statistical information while performing the primary task of achieving a converged deterministic CARP solution. The user has complete control over the sample size of the statistical solution. The CARP process indicates the sample size as it is being generated.

The en-route process does necessary processing during the traverse to the DZ to provide status information relative to both the CARP and desired release conditions. This processing includes: Prediction of the time and state of release; computation of the release time; position and velocity metrics. In a future planner upgrade, the en-route process loop can support the use of carrier way-points (computing similar time, position and velocity metrics to the next way-point).

The executive process provides control of the CARP and en-route process loops. It initiates the cyclic execution of these two processes following the initial data load. It also has the responsibility of buffering input and output data to ensure the integrity of the input data utilized by the CARP and en-route process as well as the consistency of the data that is output.

The WindPADS processor executes upon user request and obtains, stores, and applies the wind and associated information saved on file for use by the en-route process and the simulation.

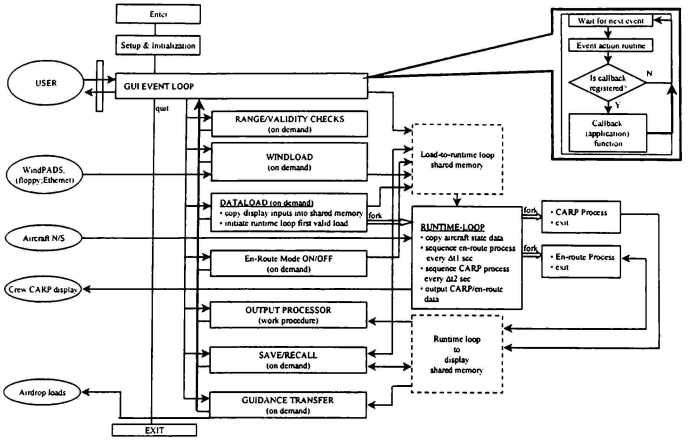


Figure 4. Planning Tool Functional Diagram

The glide trajectory processor executes on demand. It extracts the data that is required by a guided airdrop system from the CARP process. That reference descent trajectory data can then be transferred to guided airdrop systems via an output data interface that is available when a guided airdrop system is being utilized.

The user CARP interface provides for display of CARP-related information and steering directives for use by cockpit flight crew to achieve accurate load release. This information is also displayed for monitoring of planner activity.

The aircraft navigation interface provides the data for real-time state and flight data determination. The primary data source is the aircraft navigation system via a 1553 data bus (read-only) interface. Backup data sources are a stand-alone Global Positioning System receiver via a laptop serial port and manual state data entry via the user interface.

The WindPADS interface provides access by the planner to high fidelity wind data. WindPADS also supplies atmospheric (density) and terrain information around the DZ plus descent wind field information.

The planner computes and makes available for output the following information to facilitate a timely and accurate airdrop payload release:

- Release point position
  - Geodetic latitude
  - Geodetic longitude
- Release altitude metrics
  - Pressure altitude (altitude with the altimeter setting of 29.92 in Hg)
  - Ground altitude (above the highest elevation within the DZ)
  - Indicated pressure altitude (using the DZ altimeter setting)
  - Mean sea level altitude
- Release heading (to keep the aircraft on the desired course)
- Release time metrics
  - Time-to-go to release (to "green light")
  - Time-to-go to terminate release (to "red light")
- Release position metrics
  - Range-to-go to release
  - Altitude error (positive indicates the aircraft is higher than the desired CARP value)
  - Crosstrack (positive indicates the aircraft is to the right of CARP position)
- Release velocity metrics
  - Track error (positive indicates the aircraft course is more easterly than desired)
  - Speed error (positive indicates the aircraft speed is greater than desired)

All planner outputs are referenced to the initial payload release, except for time-to-go to terminate, which is referenced to the final deployment. Additionally, the planner outputs the following "advisory" information:

- Estimate of the DZ elevation

- Altimeter setting at the DZ elevation
- Projected variance in the release time from the desired (input) value

The onboard planner derives the desired output parameters using the following inputs:

- Mission objectives
  - Deliver specified number of bundle(s) to desired target(s)
- Airdrop characteristics
  - Aircraft type
  - Airdrop method (rear ramp exit or door exit)
  - Flight station at release for each load
  - Load type, weight and center of gravity (CG) height position
  - Parachute type
  - Actuation altitude and disreefing mechanism activation time delay for HALO chute types
- Release point specifications
  - DZ location, elevation, type and size
  - Number of bundles to release on run-in, and stick size
  - Drop altitude with respect to either the indicated release altitude (with aircraft altimeter set to the DZ altimeter setting) or the above ground altitude (relative to the highest terrain elevation in the DZ)
  - Indicated drop speed
  - Run-in course (the aircraft ground track heading) and distance
  - Release date and time
- WindPADS terrain and predicted atmospheric conditions in vicinity of the DZ
  - Terrain elevation
  - Atmospheric density and pressure
  - The wind field (x,y,z components) as a function of altitude, latitude, longitude, and time
- The carrier aircraft real-time state information as determined from the aircraft navigation system
  - Time, position, velocity, acceleration
  - Aircraft attitude and angular rates

The signal flow between the planner processes and external data sources is illustrated in Figure 5. Seven processes are shown: The CARP process (the CARP controller, the simulation, and the CARP algorithm); the en-route process (the en-route controller, the state propagator, and the steer-to-CARP algorithm); the wind data loading processor; the glide trajectory processor; the range-validity checks; save-recall; the GUI. The CARP process receives target airdrop specifications from the user via the GUI. The CARP process uses the simulation and CARP algorithm to update the desired payload release state. The en-route controller receives run-in specifications via the GUI as well as real-time state data from the aircraft navigation system. This information is used to propagate the state to the desired release condition derived in the CARP process. The steer-to-CARP algorithm is the computation of the CARP-relative metrics defined above. The CARP process and en-route processes receive wind data, terrain data, and atmospheric data from the wind data loading processor. All outputs pass through the output processor for formatting and consistency. Range-validity checks and save-recall functions interact with the data to/from the user by way of either the GUI or output processor. Finally the glide trajectory processor transfers reference trajectory data from the CARP process to guided airdrop systems.

THE HIGH FIDELITY WIND MODELS

The NWV-PAD airdrop planner and the embedded simulation accept quasi-static wind data formats to support both 3D and 4D models. Most airdrop simulations either assume a constant (ballistic) wind velocity during descent or provide only 2D (horizontal) wind components as a function of altitude. A horizontal wind field is not an accurate representation in situations with rugged/complex terrain, moderate to strong wind speeds, and vertically unstable atmospheric stratification. These complex flows are forced/diverted by the terrain. These same conditions produce vertical wind velocities that significantly impact the descent rate of the parachute load. In this atmospheric boundary layer (BL) the horizontal and vertical components of the wind vary significantly in 3D, and must all be modified for precise descent trajectory calculations. The BL can have a depth of up to 3 km.

Atmospheric forecast models predict the mean state of the wind at grid points that average over the time-step and grid-scale used in the model. Around the mean state, there is a turbulent component that can be represented statistically. BL models include predictions of Turbulent Kinetic Energy (TKE) along with the mean state of the wind. The variance of the 3-component wind predictions can be estimated directly from TKE (assuming isotropic turbulence). Dynamic atmospheric models also predict temperature, pressure and dewpoint temperature at each grid point. From these predictions, pressure-altitude and density can be computed directly. The PSI WindPADS system provides a full 3D wind data set as a function of the location relative to the DZ. Wind variance, density, and pressure altitude are provided over the 3D grid covering the DZ. The predicted atmospheric fields are updated, through assimilation, with in-situ measurements taken near the DZ using a dropsonde ejected from the carrier aircraft. In the current WindPADS design, wind variance (to support Monte Carlo dispersal footprint simulations) is not updated by in-situ dropsonde observations near the DZ. WindPADS enables assimilation of in-situ wind data from other systems, as

available (e.g., aircraft navigation system, airborne LIDAR, airborne doppler radar).

Atmospheric field data for the CSDL airdrop planner are determined using a dedicated PC aboard the carrier aircraft. The result is an updated, high-fidelity 3D model in the vicinity of the DZ. Furthermore, by using the mesoscale atmospheric model prediction features, the predicted wind changes over time result in a 4D (position and time-dependant) data set that brackets the planned airdrop time, and can be used to adjust for operational differences in planned vs. actual time over the DZ. WindPADS provides a formatted atmospheric data set according to a standardized interface for transfer to the CSDL airdrop planner. WindPADS components and interfaces to the CSDL airdrop planner are shown in Figure 6.

ON BOARD OPERATION STRATEGY

An operations flight strategy has been developed using the planning tool and WindPADs with each hosted on separate laptop PCs, and with an interface for transfer of the wind and atmospheric data from the WindPADs PC to the planning tool PC. In this approach, the planning tool PC has a standard (read-only) 1553 interface with the aircraft navigation system. In a

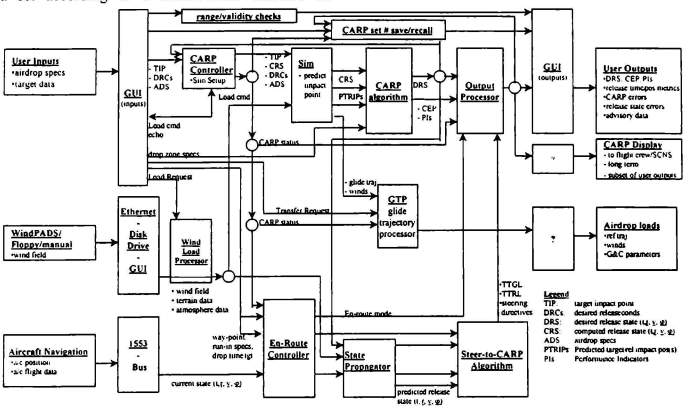


Figure 5. Planning Tool Data Flow

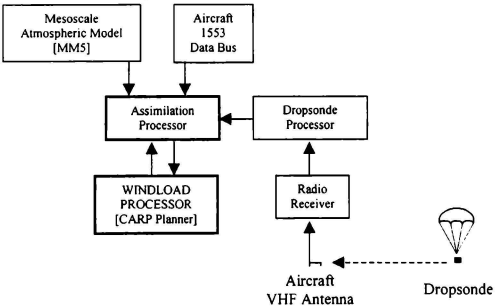


Figure 6. WindPADS Data Flow

future upgrade, both the planner and WindPADS could be hosted on a single PC. Also in a future upgrade, the planning tool PC could have an interface to drive a cockpit display that would present CARP release data and approach information directly to the cockpit crew.

The timeline associated with planner use has four phases: Pre-flight ground procedures; en-route; slowdown and release; post-release. Checklists have been developed for each of these segments for flight demonstration and nominal operations. For operations, the planning tool and WindPADS will follow an "amplified-airdrop checklist" to orient the planner and wind processing timeline requirements to existing airdrop procedures. A unique feature of the flight-demonstration is a dry-run deploy. In this scenario, the planning tool outputs are used to vector the aircraft to the desired release point and flight conditions following the nominal deployment timeline checklist. This scenario will help to evaluate steering cues, the CARP solution, and the green/red-light cues. Additionally, during this pass, airborne sensors will be released to evaluate the WindPADS reception and assimilation of the sensor data into processing, as well as to verify the transfer of the WindPADs data to the planning tool in a flight environment. Assuming that these dry-run operations meet expectations, then the aircraft will circle back to repeat the run-in to the CARP followed by release of the airdrop payloads.

During the pre-flight procedures the airdrop system characteristics, anticipated drop parameters and drop-zone conditions are entered into the planner. The high resolution forecast atmospheric fields are also entered into WindPADs. WindPADS processing is inititated to derive an atmospheric database about the DZ at the expected drop time. Following WindPADs processing and transfer its data to the planner, the planner processing is activated to derive the initial CARP solution estimate. This includes the Monte Carlo function to estimate the payload landing dispersal footprint. If the footprint, or the CARP (due to ground threat or airspace restrictions) are unsatisfactory to the operator, footprints and CARPs would be computed for different DZ times. A CARP for the dropsonde is determined in order to assure that the dropsonde probes the BL over/near the DZ.

During the en-route phase, at a predetermined time prior to release, the planner input set is reviewed, updated as required, and re-loaded. The en-route planner processing mode is then started. This transition initiates the planner's use of aircraft flight data, computation of time-to-green-light (TTGL), and release condition error metrics and steering directives. Using the same procedures as for the delivery of the main load(s), the aircraft is steered to the dropsonde CARP, the dropsonde is released/ejected, and the in-situ measured atmospheric profile is received and processed. This data and any other new meteorological data is assimilated into WindPADS, and its operator advises the planner operator when atmospheric data updates are available.

WindPADS will have two basic modes. For relatively flat terrain, the dropsonde data is assumed horizontally uniform, and merely replaces the forecast 3D field. For complex terrain, the dropsonde data is assimilated with the predicted fields and the result made available to the planning tool. As a backup, the predicted fields will be used, and if the predicted wind fields were not received, then the planner will use fields determined by the dropsonde, regardless of terrain.

The planner operator may update any inputs at any point en-route to the drop zone if the mission is modified or new atmospheric data over the drop zone is received. The planning tool will revise the CARP solution, the release condition error metrics, and steering directives based on the new input data and aircraft flight data. This en-route processing will be initiated for the demonstration prior to reaching the specified run-in distance. The run-in distance defines the minimum distance from the drop zone at which the carrier aircraft has achieved the intended course, but not necessarily the desired altitude and speed. If the carrier aircraft's approach to the drop-zone is not straight in, the planning tool will assume the vehicle uses coordinated bank turns to compute TTGL and the CARP error metrics. With a future update, the planning tool could use way-points, that define the run-in distance and the "transition point" to the desired release altitude and speed. This would allow en-route processing to be initiated sooner than in the current implementation.

During the slowdown/release phase most of the data for the planner remains fixed except for instrument calibrations such as the altimeter setting. The payloads are prepared for release from the rear deck by the loadmaster. For guided payloads the transfer of reference trajectory data from the planning tool to the individual airdrop systems is executed. Meanwhile, the flight crew brings the aircraft to the desired airdrop speed and altitude. Note that the carrier aircraft is on the final intended course heading when the slowdown occurs. The planner operator monitors the TTGL and CARP error metrics, and confirms the projected location of the aircraft relative to the CARP. At a predetermined point out from release, the airdrop would be waved off if the projected payload miss distance is

excessive. Otherwise, the release proceeds, using the planner countdown cues.

After payload release, the aircraft speed and altitude are returned to pre-deployment levels per "completion of drop" checklist procedures. For multiple airdrop passes, the planning tool and airdrop procedures are returned to the en-route processing phase in preparation for the next airdrop pass. A possible enhancement is to attach dropsonde sensors to the payloads to receive updated atmospheric information for WindPADS use prior to the next airdrop pass.

PROGRAM STATUS AND CONCLUSIONS

The conversion of the PGAS simulation to a PC-based C-language ballistic airdrop simulation has been completed, thereby creating the capability to produce CARP predictions using detailed models of the airdrop system and carrier aircraft dynamics. Models of the G-12 parachute and a cross parachute are included in the baseline simulation, with representations of the aircraft release, pre-deployment descent, and post-deployment descent all addressed in the models. The airdrop planner architecture has been defined and its PC-based C-code implementation and integration with the planner has been started. Included in the planner are capabilities to support in-flight update of the CARP and means to evaluate likely trajectory and landing footprint dispersions. Interfaces have been established in the planner to receive high fidelity atmospheric model data and to output reference trajectories for guided airdrop systems.

The capability has been developed to use data from MM5 files to build a 3D atmosphere model that supports an interface with the airdrop planner. This capability will be integrated into the PC-based WindPADS and tested (ground and airborne) with the airdrop planner on a C-130 aircraft. Testing will include in-flight dropsonde releases followed by signal ingest and onboard processing of the dropsonde data to produce in-situ atmospheric measurements.

The WindPADS and airdrop planner will undergo an airborne demonstration to establish proof-of-concept of onboard precision airdrop planning and to evaluate resulting performance capabilities. Aircraft crew recommendations from test and demonstration flights will be used to identify updates needed to convert the NWV-PAD prototype to an operational system.

The NWV-PAD program is well on the way to successful development of an airdrop carrier aircraft onboard capability to plan and execute precision airdrops from high altitude through the combined use of the CSDL airdrop planner and the PSI WindPADS system. In addition to enabling improved payload delivery accuracy they provide the onboard ability to evaluate expected airdrop performance and to replan airdrops based on changes in the mission timeline and updated wind information acquired in-flight. The flight demonstration of these systems onboard a C-130 will occur in 2001.

REFERENCES

1. S. Dellicker, I. Kaminer, R. Howard, R. Benney, and S Patel, "Performance, Control and Simulation of the Affordable Guided Airdrop System," AIAA paper 2000-4309, presented at the AIAA Guidance, Navigation, and Control Conference, August 14-17, 2000, Denver, Colorado.
2. G. Brown, R. Haggard, R. Benney, and M Gawronski, "Parachute Retraction Soft-Landing Systems Using Pneumatic Muscle Actuators," AIAA paper 2000-4308, presented at the AIAA Guidance, Navigation, and Control Conference, August 14-17, 2000, Denver, Colorado.
3. K. Stein, R. Benney, T. Tezduyar, M. Accorsi, and J. Leonard, "Fluid-Structure Interaction Modeling of the US Army Personnel Parachute System," AIAA paper 2000-4310, presented at the AIAA Guidance, Navigation, and Control Conference, August 14-17, 2000, Denver, Colorado.
4. P. Hattis, B. Appleby, T. Fill, and R. Benney, "Precision Guided Airdrop System Flight Test Results," AIAA paper 97-1468, presented at the AIAA Aerodynamic Decelerator Systems Technology Conference, June 3-5, 1997, San Francisco, California.

ACKNOWLEDGEMENTS

The authors wish to thank Dr. Steven Walker of the Air Force Office of Scientific Research in Arlington, Virginia for his consistent as well as strong support and direction of the New World Vistas Precision Aerial Delivery Program. The C. S. Draper Laboratory portion of the work discussed in this paper was sponsored under contract DAAN02-98-K-0004-P00004 with the U.S. Army Soldier and Biological Chemical Command. Publication of this paper does not constitute approval by Army or the C. S. Draper Laboratory, Inc. of the findings or conclusions contained herein.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37300

AIAA-2000-4308

PARACHUTE RETRACTION SOFT-LANDING SYSTEMS USING PNEUMATIC MUSCLE ACTUATORS

Glen Brown* and Roy Haggard†
Vertigo, Inc.
P.O. Box 117
Lake Elsinore, CA 92531-0117

Richard Benney‡
US Army Soldier & Biological Chemical Command
Soldier Systems Center
Natick, MA 01760-5017

ABSTRACT

Newly developed flexible composite technology makes compact and highly efficient long-stroke actuators for parachute retraction possible. These actuators are slender flexible tubes with a mechanical attachment at each end and a pneumatic fitting at one end. Prior to pressurization, the actuator is flexible and can be packed with the parachute. When pressure is applied through the end fitting, the actuator contracts strongly, providing a force useful for a variety of applications. The name "Pneumatic Muscle Actuator" (PMA) is descriptive of this device.

This paper discusses parachute retraction and the use of PMAs for the soft landing of cargo and vehicles. Parachute retraction soft landing is currently being studied at the US Army Soldier & Biological Chemical Command, Soldier Systems Center where continuing analysis and tests are showing this concept's potential. This paper discusses the characteristics of PMAs as they apply to soft landing and progress toward implementing a practical soft-landing system for cargo.

SOFT LANDING SYSTEM DESCRIPTION

A soft-landing airdrop system consists of a payload, parachute(s) and the retraction device. The payload and the parachute can be completely standard, inventory items. In the example of an airdropped HMMWV (Figures 1 and 2) we assume a payload weight of approximately 20,000 lb. and the use of four G-11 parachutes. This is an interesting example and an application with high potential payoff because this combination descends at approximately 28 ft/sec, but could land at 10 ft/sec or less eliminating the need for impact attenuation material.

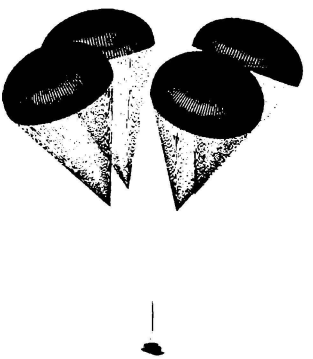


Figure 1 Airdropped HMMWV

* Director of Engineering, Senior Member
† Engineering Manager, AIAA Member
‡ Aerospace Engineer, Senior Member

This paper is declared a work of the US Government and is not subject to copyright protection in the United States.

A pneumatic muscle actuator (PMA) used as a retraction device consists of a braided fiber tube, a gas barrier liner, one end closure with a mechanical load attachment, and one end with a mechanical load attachment, a pyrotechnic gas generator and a ground proximity sensing initiator.



Figure 2 Airdropped HMMWV

PMA FORCE EQUATION

The actuator is constructed as a tube of braided fibers. The braid includes only bias fibers, which allows the tube to change length in response to force and pressure changes. The actuator also changes diameter and fiber bias angle whenever length changes in response to pressure or applied load.

It is convenient to express the tension generated by the PMA as a coefficient normalized on the force generated by a pneumatic cylinder of the same initial diameter.

$$C_T = \frac{F}{\frac{\pi}{4} P d_o^2}$$

The force coefficient is given below, as derived in reference 7. l_o and β_o are the initial length and bias angle respectively.

$$C_T = \frac{\left[\sin^2\left(\cos^{-1}\left(\frac{l}{l_o}\cos\beta_o\right)\right) - 2\cos^2\left(\cos^{-1}\left(\frac{l}{l_o}\cos\beta_o\right)\right)\right]}{\sin^2\beta_o}$$

Maximum stroke and force is produced by an actuator having a small initial bias angle. The graph found in Figure 3 was calculated for a pneumatic muscle actuator having an initial bias angle of 15 degrees. Note the remarkably high tension coefficient, starting at well over twenty, and the useful contraction of nearly 40 percent.

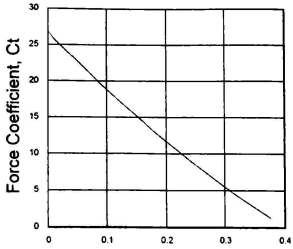


Figure 3 PMA Force Curve from Theory

Actual force generated is reduced by any resistance of the individual fiber bundles to pivoting on one another with bias angle changes and by the expansion of the stretched elastomeric gas barrier inside the braided tube. The actual force is also reduced by the fixed end fittings, which are not allowed to expand in diameter as does the rest of the tube. The following section compares static force measurements with this simplified estimate.

STATIC TESTING

Prototype fabrication and testing were used to validate the predicted PMA performance. The graph contained in Figure 4 shows the force generated in a 1.05 inch diameter, 34 inch long PMA.

Actual force generated is less than the theoretical prediction. This is to be expected from end effects, liner elasticity and fiber-fiber friction. The ratio of actual to predicted force is also plotted. Over the operating range, the efficiency of the muscle is in the range 85% to 65%, with the highest efficiency during initial contraction.

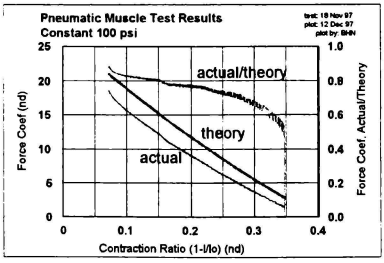


Figure 4 Measured Force Compared to Theory

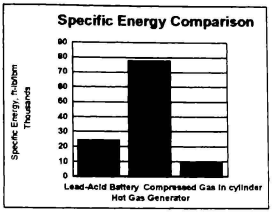
PERFORMANCE COMPARISONS

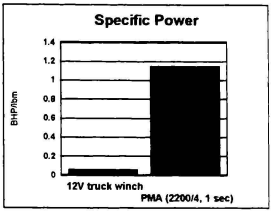
How do PMAs compare to other devices capable of the same action? Figure 5 illustrates several comparisons.

Inflating PMAs requires the use of either stored compressed gas or a pyrotechnic gas generator. The first bar graph compares an advanced lead-acid battery with a hot pyrotechnically generated gas (1000°F) and a compressed gas stored at 59°F. When the work required is of short enough duration to allow the use of a single discharge of hot gas, the pyrotechnic generator is superior. For multiple actuator strokes over a long period of time, the lead-acid battery is superior when compared to compressed gas storage.

Specific power is compared in the second bar graph of Figure 5. A truck winch is used as an example of an electromechanical device. The PMA is far superior because increasing the power of a PMA requires only increasing the size of the flow passage, which does not increase the weight of the PMA. Increasing the power of an electric motor requires additional iron and copper in direct proportion to the amount of power required. This comparison is applicable to parachute retraction, parafoil steering and parafoil flare.

The third comparison shown in Figure 5 is for the specific impulse, which is most useful for soft landing retraction. The PMA is compared to a solid propellant retro-rocket for which a good specific impulse would be approximately 280 seconds. Various simulations of PMA retraction processes indicate a specific impulse of approximately 1400 seconds, which represents a considerable improvement.





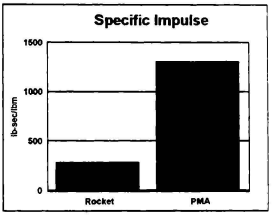


Figure 5 PMA Performance

SOFT LANDING PERFORMANCE

Krainski[1] has previously reported on the simulation of a soft landing retractor implemented with a mechanically stroke-amplified pneumatic cylinder. The primary differences in the simulation of this system when compared to Krainski's work are in the number of rigid bodies and the flow model. Because the majority of the mass of the device moves with either the payload or the parachute, a 2-body simulation is adequate for the pneumatic muscle. Krainski uses a volume of stored gas that communicates with no flow restriction to the pneumatic cylinder at the time of initiation, while the PMA system under investigation uses choked flow from a hot, high pressure source.

A simple simulation was written in spreadsheet format to explore the design variables and performance. The integration method is stepwise constant acceleration, using the forces of the previous time step to calculate the displacement and velocity. This allows calculation in a single pass through the sheet, is accurate for slowly varying forces, and can be verified by looking at the sensitivity of the result to the time step value.

The graphs in Figure 6 show performance estimates for a 14 ft long, 6.0 inch diameter PMA retracting a 10,000 lbm payload. A gas generator burn of 6.0 lb/sec for 0.20 seconds produces a peak acceleration of 8.8 g and a ΔV of 18.8 ft/sec.

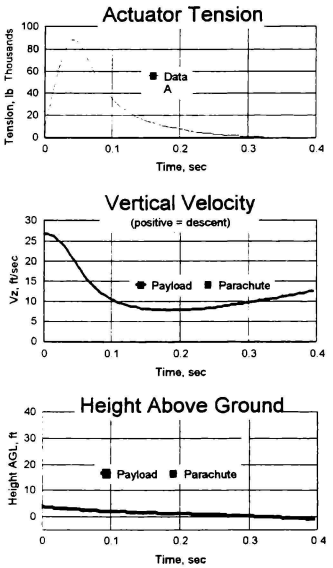


Figure 6 Calculated Soft Landing of 10,000 lbm Payload

With an initial descend rate of 26.7 ft/sec (under two G-11 parachutes), descent rate is reduced to less than 10 ft/sec for a period of 0.21 seconds. If the trigger initiates at between 3.7 and 1.9 (2.8 and 0.9) feet above the ground, the resulting impact will be less than 10 ft/sec.

DYNAMIC TESTING

A dynamic test was also conducted using a 300 lb torso dummy and a 1.05 inch diameter, 16 ft long PMA. The test setup is shown in the photo of Figure 7, with the dummy at the top of its trajectory, pulling against a bracket on the side of a building.

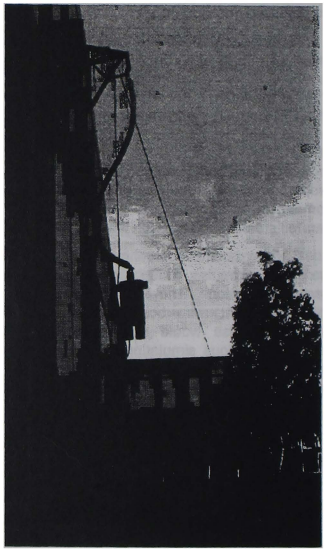


Figure 7 Dynamic Test with 300 lbm

The dummy was lifted off the ground by rapid inflation from a reservoir tank that had initial pressures varying from 25 to 150 psig. The flow into the PMA was slowed by pipe losses compared to an integrated gas generator, but the contraction was still strong and rapid. PMA tension loads were measured and integrated to derive the graph of velocity vs time shown in Figure 8.

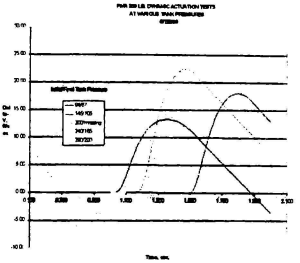


Figure 8 Velocity vs. Time

Parachute drop tests were conducted with the same PMA and suspended weights of 222 lb. and 305 lb.. The payload consisted of the high pressure tank with ballast weights shown in Figure 9. Note in the photograph the trigger rod attached to the side of the tank. The rod opens the valve at the top of the tank when it contacts the ground.

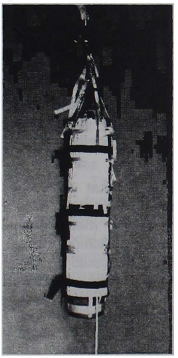


Figure 9 High Pressure Tank

This setup was dropped under a T-10 parachute from a drop tower at Ft. Benning. (Figure 10). A load cell between the payload and the PMA was recorded. From the force vs time data, peak force, peak acceleration and delta-V were derived. The results are plotted in Figure 11 and summarized in Table 1.

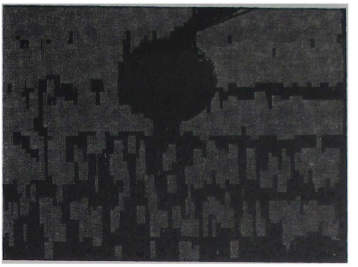


Figure 10 Drop Test

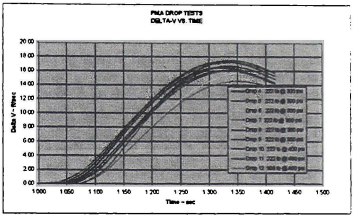


Figure 11 PMA Drop Test Plot

DROP NO.	WEIGHT Lb.	PRESSURE Psi	PEAK FORCE Lb.	PEAK G	DELTA-V max ft/sec
4	222	300	880	2.97	16.5
5	222	300	848	2.83	16.1
6	222	300	804	2.63	15.8
7	222	300	850	2.84	17.2
8	222	350	874	2.95	17.4
9	222	350	882	2.98	16.7
10	222	400	846	2.83	16.1
11	222	400	942	3.25	17.5
12	305	400	1065	2.78	14.5

Table 1 PMA Drop Test Summary

Even though the conditions of payload weight and initial pressure were not duplicated between the fixed-end tests and the parachute drops, it is interesting to note that the delta-Vs of approximately 25 ft/sec with the fixed end were reduced to approximately 15 ft/sec with the parachute. This is a result of the parachute accelerating toward the payload mass during retraction. We hope, in the future, to have good enough data to infer the parachute's effective mass (included mass plus apparent mass) from tests such as these.

FINITE ELEMENT MODELING OF PMAS

The ability to predict PMA performance and behavior with a high fidelity model is desirable for the prediction of forces applied to the load, the effects of end caps on performance, varying material properties, static shapes under loads, force histories, and other real effects not included in the basic force equation.

A finite-element-based structural dynamic (SD) model is utilized to model PMA statics and dynamics. The software known as "TENSION9" has many features that make it especially useful for modeling fiber reinforced membrane structures like PMAs. The TENSION9 theory with airdrop system modeling examples are described in detail in other works[3,4]. TENSION9 is also being applied to model controllable round and cross canopy airdrop systems where a PMA control actuator could deform the canopy system and produce glide[6]. The code is coupled to a pre and post processing software suite written in Matlab$_{TM}$. A module to "build" PMA models has been constructed for easy assembly of PMA "unit cells" and duplication of the unit cells into PMA structures.

In addition, the TENSION9 software has recently added a user-defined, rotated-coordinate-system feature which allows the user to build a single column of PMA unit cells to model a "sliver" of the PMA's circumference in detail down to every fiber in the braid but at significantly lower computational cost. Simulations with this option are ongoing and will not be reported in detail here. A one-inch diameter, 15-foot long, quarter-symmetry PMA model is presented here as an example (Figure 12).

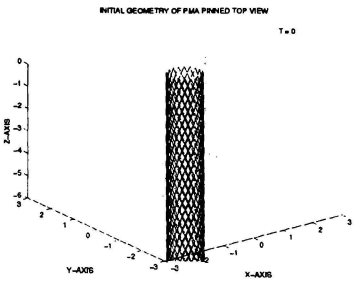


Figure 12 FEA model of PMA

For this 1 inch diameter model, the overall length is 15 feet (close to the pull up experiments) and the weight applied to the base is equivalent to a total payload weight of 250 pounds. In this case, the model's circumference contains 5 unit cells while the length is 307 unit cells long. The total model contains 1535 unit cells. This equates to 3,383 node points and 12,286 elements (6,140 two-noded cables, 6,140 3-noded membranes and 6 concentrated mass elements). The model is initially unstretched and both an internal pressure of five psi and gravity are "turned on" at the start of the simulation. The pressure is increased from five psi to 50 psi over the first 0.1 seconds and then held at 50 psi for the duration of the simulation. A small quantity of mass-proportional damping is used throughout the simulation. This is used to assist in numerical stability and simulate a small amount of damping in the fiber braid material which may be modeled with two-noded dampers in the future.

Additional features that may be added include kink and/or fold elements applied to model the friction effects of the membrane/fiber and fiber/fiber interactions. In addition, utilization of a membrane wrinkling model could be applied to the inner liner membrane (see ref 5 for more information on specialized features of TENSION9). The simulation is run for five seconds in real time.

The time history of a payload node and velocity in the vertical direction are shown in Figure 13. The simulation predicts a damped oscillation as expected with the first "peak" being of primary interest in this simulation. The model predicts a payload vertical pull of nearly six feet (over 70 inches) during the first peak. The other nodes follow the same trend as expected and witnessed during the real tests. The velocity of the payload reaches well over 11 feet/second (over 130 inches/second).

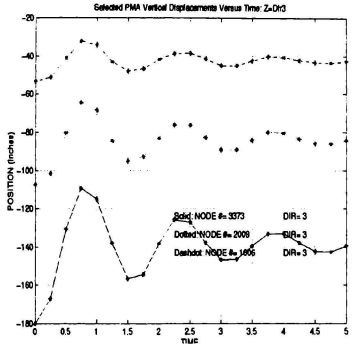


Figure 13 FEA dynamic results

The end cap shape at the top of the PMA is shown in Figure 14 at a time of five seconds (the final time of the simulation). The shape appears as expected based on static tests conducted with the one inch diameter PMA.

These results are relatively close to those obtained experimentally considering the crude material properties utilized and the simplified loading applied. The model also predicts the accelerations (G-forces) applied to the payload, the time dependent forces experienced by the fibers in the model and the stresses experienced in the liner membrane. The model has many assumptions including: a) The membrane and fibers are attached to each other and relative sliding is not included, b) The fiber braids are "pinned" to each other throughout the simulation, c) For the symmetry models, only symmetric effects are modeled (full 3D models can be used but they require more computational time), d) The frictional effects are approximated, e) The applied pressure is linearly ramped to a constant level and is kept constant during the oscillations, f) The material properties are approximated and assumed to be linearly elastic (the fiber Young's Modulus has a significant effect on results).

Continued simulation of PMAs is ongoing. A "wedge" model utilizing higher order PMA unit cells has been constructed to model the large six inch diameter PMA pull tests and static tests.

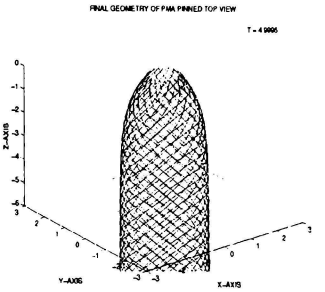


Figure 14 FEA geometry at one PMA end

These models are expected to assist engineers in designing and utilization of PMAs. As demonstrated, end cap effects can be modeled. PMA simulations should assist in the prediction of changing material types (braid material or liner material), dimensional and scaling changes, soft-end-cap designs, maximum effective loading and the effect of over-pressure on PMA stiffness, PMA responses to limited or no internal pressure, and many other features. The TENSION9 software suite has demonstrated the ability to model PMA structures and this modeling will continue as the program develops. The software can be applied for predicting the dynamic and static behavior of many fiber reinforced membrane structures.

LARGE MUSCLE TESTING

A large, 6-inch-diameter PMA has been constructed for testing (Figure 15). The purpose is to demonstrate the manufacturing technology needed for large PMAs and the force generating capability needed for soft landing of vehicles. Force in excess of 100,000 lbf is expected from this test article. Tests were conducted using a large crane at the U.S. Army Soldier Systems Center (Natick) and are continuing through parachute airdrops this Fall (2000).

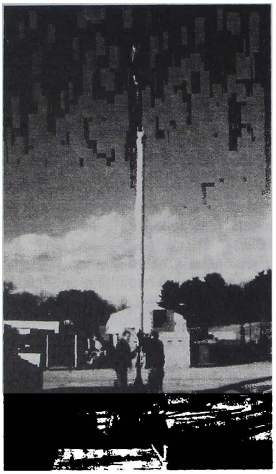


Figure 15 Large Muscle Test Setup

CONCLUSIONS

Pneumatic muscle actuators (PMAs) have a high potential for use in advanced recovery systems. Applications may include the following:

- Landing impact reduction for cargo and personnel;
- Parafoil steering and flare; and
- Parachute trajectory control.

A force equation has been derived for the ideal PMA and validated by static testing. Static testing also provides a measure of the efficiency of actual PMAs. Finite element analysis is being applied to develop a more general tool for predicting performance with elastic resistance and end effects.

Substantial velocity reduction has been demonstrated with parachute drop tests. The feasibility of applying this approach to heavy cargo has been shown by the construction ans testing of larger pneumatic muscle actuators. Tests planned for the fall of 2000 will demonstrate landing velocity reduction with parachute drops of over 10,000 lb.

REFERENCES

1. W.J. Krainski and S.E. Kunz, "The Dynamics of a Parachute Retraction Soft Landing System", Proceedings of the 14th AIAA Aerodynamic Decelerator Conference, San Francisco, CA, 1997.

2. G. Brown, R. Haggard, R. Almassy, R. Benney, S. Dellicker, "The Affordable Guided Airdrop System", Proceedings of the CEAS/AIAA 15th Aerodynamic Decelerator Systems Technology Conference, AIAA-99-1742, Toulouse, France, 1999.

3. R. J. Benney & J.W. Leonard, "A 3-D Finite Element Structural Parachute Model", Proceedings of the 13th AIAA Aerodynamic Decelerator Conference, Clearwater, FL, 1995.

4. R. J. Benney, K.R. Stein, J.W. Leonard, M.L. Accorsi, "Current 3-D Structural Dynamic Finite Element Modeling Capabilities", Proceedings of the 14th AIAA Aerodynamic Decelerator Conference, San Francisco, CA, 1997.

5. M. Accorsi, K. Lu, J. Leonard, R. Benney, K. Stein, "Issues in Parachute Structural Modeling: Damping and Wrinkling", Proceedings of the CEAS/AIAA 15th Aerodynamic Decelerator Systems Technology Conference, AIAA-99-1719, Toulouse, France, 1999.

6. R. J. Benney, K. Stein, W. Zhang, M. Accorsi, J. Leonard, "Controllable Airdrop Simulations Utilizing a 3-D Structural Dynamic Model", Proceedings of the CEAS/AIAA 15th Aerodynamic Decelerator Systems Technology Conference, AIAA-99-1727, Toulouse, France, 1999.

7. G. Brown, R. Haggard, R. Benney, N. Rosato, "A New Pneumatic Actuator and Its Use in Airdrop Appliactions", Proceedings of the CEAS/AIAA 15th Aerodynamic Decelerator Systems Technology Conference, AIAA-99-1719, Toulouse, France, 1999.

A00-37301

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

AIAA-2000-4309

PERFORMANCE, CONTROL, AND SIMULATION OF THE AFFORDABLE GUIDED AIRDROP SYSTEM

S.Dellicker
US Army Yuma Proving Ground, Yuma, AZ
R.Benney, S.Patel
US Army SBCCOM, Natick, MA
T.Williams, C.Hewgley, O.Yakimenko[#], R.Howard, I.Kaminer
Naval Postgraduate School, Monterey, CA

Abstract

This paper addresses the development of an autonomous guidance, navigation and control system for a flat solid circular parachute. This effort is a part of the Affordable Guided Airdrop System (AGAS) that integrates a low-cost guidance and control system into fielded cargo air delivery systems. The paper describes the AGAS concept, its architecture and components. It then reviews the literature on circular parachute modeling and introduces a simplified model of a parachute. This model is used to develop and evaluate the performance of a modified bang-bang control system to steer the AGAS along a pre-specified trajectory towards a desired landing point. The synthesis of the optimal control strategy based on Pontryagin's principle of optimality is also presented. The paper is intended to be a summary of the current state of AGAS development. The paper ends with the summary of the future plans in this area.

I Introduction

The United States Air Force Science Advisory board was tasked to develop a forecast of the requirements of the most advanced air and space ideas to project the Air Force into the next century. The study, encompassing all aspects of Air Force operations, assessed a variety of technology developments critical to the Air Force mission. This study culminated in a report titled "New World Vistas, Air and Space Power for the 21st Century."[1] The study identified a critical need to improve the Point-of-Use Delivery; that is, getting the materiel where it needs to be, when it needs to be there. Airdrop is an important aspect of Point-of-Use Delivery. The report indicated that immediate improvements are needed with emphasis provided by the statement: "In the future, the problem of airdrop should be treated as seriously as the problem of bomb drop."

The first attempts to develop such systems are as old as the introduction of gliding, maneuverable parachutes.[2] However, practical systems had to wait for the development of hi-glide parachutes, especially the ram-air inflated parafoil. In 1969 the US Army defined requirements for and discussed such a cargo point delivery system.[3] None of attempts in the 70's and 80's to develop such a system were operationally acceptable, however nowadays such systems have been developed (see for example Ref.4).

These large-scale parafoil systems use a marker or beacon on the ground and ensure 99% landing accuracy in a hundred-yard circle around the beacon. Therefore, they provide the accuracy required with delivery from high altitude and large offset distances. The drawback is prohibitive cost for each pound of payload delivered. Alternate approaches were required to reduce system cost. Improved Affordable Airdrop Technologies are being evaluated by the team of the US Army and Air Force, the Naval Postgraduate School, The Boeing Company, and Vertigo, Incorporated. These efforts include the design and development of the AGAS, which incorporates a low-cost guidance, navigation, and control system into fielded cargo air delivery systems. This study focuses on evaluating the feasibility of the AGAS concept and encompasses the design and execution of a flight test program to assess dynamic response of a flat circular parachute, the design of initial guidance and control techniques, and to evaluate the feasibility of the AGAS concept.

II AGAS concept, architecture and components

AGAS is being evaluated as a low-cost alternative for meeting the military's requirements for precision airdrop.[5,6] Designed to bridge the gap between expensive high glide parafoil systems and uncontrolled (ballistic) round parachutes, the AGAS concept offers the benefits of high altitude parachute releases but cannot provide the same level of offset from the desired impact point (IP) as high-glide systems. The design goal of the

[#] National Research Council Senior Research Associate at the NPS.

This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AGAS development is to provide a Guidance, Navigation, and Control (GNC) system that can be placed in-line with existing fielded cargo parachute systems (G-12 and G-11) and standard delivery containers (A-22). The system is required to provide an accuracy of 100*m*, Circular Error Probable (CEP), with a desired goal of 50*m* CEP. No changes to the parachute or cargo system are allowed.

The current design concept includes implementation of commercial Global Positioning System (GPS) receiver and a heading reference as the navigation sensors, a guidance computer to determine and activate the desired control input, and the application of Pneumatic Muscle Actuators (PMAs) to effect the control. The navigation system and guidance computer will be secured to existing container delivery system while the PMAs would be attached to each of four parachute risers and to the container (Figure 1). Control is affected by lengthening a single or two adjacent actuators. The parachute deforms creating an unsymmetrical shape, essentially shifting the center of pressure, and providing a drive or slip condition. Upon deployment of the system from the aircraft, the guidance computer would steer the system along a pre-planned trajectory. This concept relies on the sufficient control authority to be produced to overcome errors in wind estimation and the point of release of the system from the aircraft. Following subsections discuss main AGAS components.

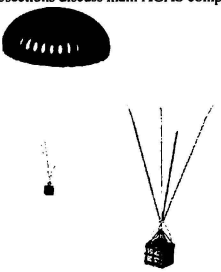


Figure 1. Affordable Guided Airdrop System#

For an airdrop mission, the aircrew will determine the Computed Air Release Point (CARP) based on the best wind estimate available at that time. The aircraft will then be navigated to that point for air delivery of the materiel. Should the wind estimate and calculation of the predicted release point be perfect and the aircrew gets the aircraft to the precise release point, then the parachute would fly precisely to the target without control inputs. However, wind estimation is far from a precise science. The calculation of the CARP relies on less than perfect estimates of parachute aerodynamics and the flight crews cannot possible precisely hit the predicted release point for each airdrop mission. Therefore, the AGAS control system design must help overcome these potential errors.

II.1 Parachute

Until now two solid flat circular parachutes C-9 and G-12 were modeled to demonstrate a feasibility of AGAS concept. (A flat circular parachute is one that when laid out on the ground forms a circle.[2]) Figure 2 shows a deployed configuration of C-9. Although the C-9 was initially designed as an ejection seat parachute, it is a standard flat circular parachute as are the larger G-11 and G-12 cargo parachutes on which AGAS will ultimately be used. Some data on these parachutes can be found in the Table 1.

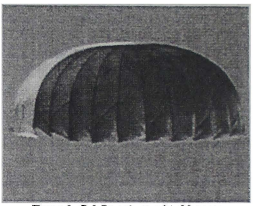


Figure 2. C-9 Parachute with 28 gores

Table 1. Parachutes data[2]

Parameter	C-9	G-12	G11-A
d_0 (ft)	28	64	100
d_P / d_0	0.67	0.67	0.67
Number of suspension lines	28	64	120
l_0 / d_0	0.82	0.80	0.90
C_{D0}	0.68	0.73	0.68
Parachute weight (lbs.)	11.3	130	215
Payload weight (lbs.)	200	2,200	3,500
Rate of descent (fps)	20	28	22

In this table d_0 denotes the nominal diameter of the parachute, d_P - inflated canopy diameter, C_{D0} - a drag coefficient, and l_0 - a suspension line length.

A cargo box is suspended from the system and houses the remote control system, control actuators, and instrumentation system.

II.2 Actuators

Vertigo, Incorporated developed PMAs[7] to effect the control inputs for this system. The PMAs are braided fiber tubes with neoprene inner sleeves that can be pressurized. Uninflated PMAs as installed on a scaled sys-

Courtesy of Vertigo, Inc., Lake Elsinore, CA.

tem are shown in Figure 3. Upon pressurization, the PMAs contract in length and expand in diameter.



Figure 3. Pneumatic muscle actuators

With four independently controlled actuators, two of which can be activated simultaneously, eight different control inputs can be affected. The concept employed for the AGAS is to fully pressurize all actuators upon successful deployment of the parachute. To affect control of the system, one or two actuators are depressurized. This action "deforms" the parachute creating drive in the opposite direction of the control action.

Figure 4 shows a diagram of the actuator setup in the parachute payload provided by Vertigo, Incorporated, the makers of the PMAs. The gas for filling the actuators comes from 4500*psi* reservoirs (the diagram shows two, but in the simulation for this study, only a single 4500*psi* reservoir is used). Each of the four actuators are then connected to this same reservoir of nitrogen gas through some piping or tubing leading to a fill valve. The fill valve is opened to allow gas to fill the actuators when a command to take an actuation off is received. When the pressure inside the PMA reaches a certain value, a pressure switch signals the fill valve to close.

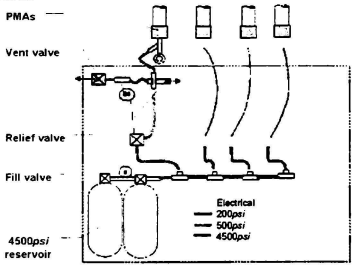


Figure 4. Vertigo, Inc. actuator system concept

Since the fill valve works with high-pressure gas it has a small orifice and therefore opens and closes rather quickly upon receiving the correct electrical signal. The time to open and close the valve is roughly 100*ms*. However, the decrease in pressure of the gas tank as more and more fills are completed slows down the actual filling process. Some of this data is plotted in Figure 5, showing increasing fill time as a function of decreasing tank pressure for actuators being filled to three different pressures.

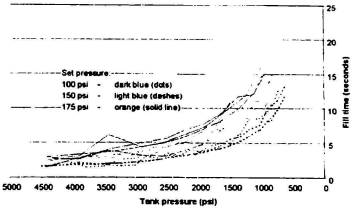


Figure 5. Fill time versus tank pressure

The vent valve opens to empty the actuator when a command to actuate is received. The vent valve has a large orifice and can open quickly to vent the PMA, but requires a certain time to vent the gas and close the orifice. The opening of the vent valve requires approximately 100*ms*, but the venting process and closing of the valve depends on the maximum pressure of the actuator fill. This process also takes a constant amount of time (approximately) because the pressure in the actuators is the same upon each vent. Figure 6 includes a diagram of the computer-modeling concept for the actuators.

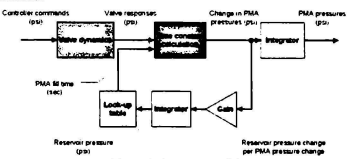


Figure 6. Actuator model

Controller commands are input to the system for each of the four PMAs. The controller commands are the pressure signals for each PMA; with 0*psi* being a vent of the actuator and 175*psi* (or the maximum pressure of the actuator) being a fill of the PMA. These commands are then passed through code that models the dynamics of the valves. This code is just a first order lag with a rise time of approximately 100*ms* to model the opening and closing of the valves. The valve responses are then passed to a code that models the actual venting and filling of the actuators, keeping in mind that the venting process takes a constant amount of time and the filling process increases with decreasing tank pressure. Once again this behavior is modeled as a first order lag. This code outputs the derivative with respect to time of the

PMA pressures. This is integrated to give the current state of each of the PMAs at a given time. PMA fill time is calculated by passing the change in PMA pressures through a gain that reflects data taken on the reservoir pressure changes per PMA pressure changes or the amount of reservoir pressure depleted for every actuator fill. This negative gain is integrated from an initial reservoir pressure to give current reservoir pressure. A look-up table is used to provide a value for the fill time of the actuators based on the remaining reservoir pressure from experimental data.

III Parachute modeling

Significant amount of research on flat circular parachute modeling has been done over past fifty years by researchers in US, Europe and Russia. It has been recognized that the basic parachute model is similar to that of a conventional aircraft. However, parachutes do have several unique features. First of all, they must be modeled as a flexible structure that includes both the parachute and the payload. Secondly, the added masses must also be modeled. And lastly it appears that no one attempted to model the control effectiveness of the risers. This section summarizes some of the research done in this area and introduces a simplified model used in this paper for preliminary analysis of control strategies.

III.1 Literature review

In 1968 White and Wolf[8] developed a 5-DOF dynamic model (ignoring the N equation, "yaw" about the downward z-axis), reviewing Henn (1944) and Lester (1962), noting that Lester discussed certain errors in the equations of motion by Henn. They noted that Lester carefully derived the equations of motion but did not attempt any solutions. They used scalar values for apparent mass and moment of inertia and conducted a dynamic stability study in which 1) steady vertical descent, 2) steady gliding ($\alpha \neq 0$, with α defined from the z axis), 3) large-angle pitch oscillation, and 4) coning are considered. A linearized analysis led to uncoupled longitudinal and lateral motions as is typical with aircraft.

It was noted that because of the shapes of the force-coefficient curves ($C_N = \sqrt{X^2 + Y^2}(qS_0)^{-1}$, the normal force coefficient, and $C_T = -Z(qS_0)^{-1}$, the tangential force coefficient), most parachutes do not fall vertically, but trim at a stable glide angle α_0. Even so, statically-stable parachutes may oscillate in pitch rather than glide. The authors noted that based on data from Heinrich and Haak[9], most parachutes could not develop sufficient $C_{N\alpha}$ to be glide stable except at high load masses, high descent rates, and low effective porosity. They noted the personnel guide surface canopy, which has low $C_{N\alpha}$ and high canopy-mass-to-payload-mass ratio, would be expected to oscillate rather than glide. Small lateral motions may result in neutrally-stable oscillations, but coning cannot develop: the longitudinal and lateral motions are uncoupled.

A nonlinear analysis produced different results. A parachute stable in the linearized sense will jump to a large-angle pitch oscillation if struck with a longitudinal disturbance greater than half of its stable angle α_0. On the other hand, a large lateral disturbance induces a longitudinal motion of comparable magnitude, throwing the parachute into a uniform vertical coning motion. To summarize: a glide-stable parachute is neutrally stable to a small lateral disturbance, but may jump into a vertical coning motion if hit with a large lateral disturbance. A single coning solution was found to exist for any given system, and only the parameters α_0 and $C_{N\alpha}$ have a great effect on coning.

Doherr[10] attempted to simulate the oscillatory behavior of a parachute and payload using two rotational DOF for the payload and three rotational DOF for the parachute. He noted that theory predicts all parachutes to assume a stable position at $\alpha = \alpha_{stable}$, while during wind-tunnel tests the less stable parachutes continued to oscillate. He recommended that aerodynamic coefficients be functions of both α and time, i.e., $C_N = C_N(\alpha(t), t)$.

Tory and Ayres[11] used a complete 6-DOF model. Differences they pointed out with White and Wolf's model are the additional DOF, forces considered on the payload, and the consideration of the apparent mass as a tensor. They showed plots of lift, drag and moment on a flat circular canopy determined from wind-tunnel tests, which will be useful for aerodynamic modeling. They modeled the apparent masses and moments of inertia based on Lamb's expressions and the assumption of the canopy being shaped as a 2:1 ellipsoid. They compared their results to some flight data, and noted that apparent inertia may not be as critical a parameter as had been suggested. A simulated drop exhibited oscillations of ±30° with a period of 5 seconds, coincident with the plane of the trajectory (therefore all pitch - no coning). Actual tests showed oscillations of 20-30° magnitude with a period of about 4-5 seconds.

Eaton and Cockrell[12] described a series of planned drop tests, from which the center of rotation was to be determined and the apparent mass estimated, but no tests were apparently conducted.

As noted by Lester and presented in the open literature by Doherr and Saliaris[13], in a 6-DOF system in which apparent mass and inertia terms are important, an apparent mass tensor can be represented by the coefficients α_{ij} to form a symmetric matrix, which for the

rigid parachute-payload system satisfies the following equalities: $\alpha_{ij} = a_{ji}$, $\alpha_{22} = \alpha_{11} \neq 0$, $\alpha_{33} \neq 0$, $\alpha_{44} = \alpha_{55} \neq 0$, $\alpha_{15} = -\alpha_{24} \neq 0$. The rest of the elements in this case are equal to zero.

Doherr and Saliaris[13] noted that predicted dynamic stability depends strongly on the math model of apparent mass. They detailed errors of Henn as pointed out by Lester. It is interesting that though White and Wolf discussed Lester's corrections of Henn's equations, they chose to use a scalar model of apparent mass and moment of inertia, rather than include the apparent-mass-tensor terms as documented by Doherr and Saliaris. Doherr and Saliaris use the set of apparent-mass terms in a small-disturbance 3-DOF model of dynamic stability. For apparent mass m_a they use half the mass of the air included within the canopy hemisphere, "arbitrarily" as noted later by Cockrell and Haidar[14]. The apparent-mass terms are given by the following equations:

$$\alpha_{11} = m_a = \frac{1}{4}\rho\frac{4}{3}\pi\left(\frac{d_P}{2}\right)^3,\ \alpha_{22} = \alpha_{11},\ \alpha_{33} = 2\alpha_{11},$$

$$\alpha_{55} = 0.048\alpha_{11}d_P^2 \text{ and } \alpha_{15} = \pm 0.2\alpha_{11}l_0. \quad (1)$$

Doherr and Saliaris considered five cases: i) $\alpha_{11} = \alpha_{33} = \alpha_{15} = \alpha_{55} = 0$ (all apparent mass terms being neglected); ii) $\alpha_{33} = \alpha_{11}$ and $\alpha_{15} = 0$ (apparent mass constant, without the coupling term); iii) $\alpha_{33} = 2\alpha_{11}$ (tangential apparent mass is twice normal apparent mass); iv-v) α_{15} is negative and positive.

The authors noted the importance of nonlinear force terms and apparent-mass effects. They referred to the Yavuz and Cockrell[15] study which showed that the apparent-mass terms are not constant, but are dependent upon the acceleration modulus (for example, $\dot{w}d_P V^{-2}$ for α_{33}).

Cockrell and Doherr[16] referred to White and Wolf as a widely accepted and employed parachute modeling method, yet lamented the lack of validation from flight test data. They aggressively pointed out that the equations of Tory and Ayres were developed incorrectly, though the particular errors were not noted. Cockrell and Doherr showed the same 3-DOF equations as Doherr and Saliaris denoting them as Lester's equations. They also listed the full 6-DOF form of the equations and referred to Eaton[17] for their derivation.

Yavuz and Cockrell[15] repeated the same 6-DOF equations as Cockrell and Doherr with the same remaining independent apparent-mass terms: α_{11}, α_{33}, α_{55}, and α_{15}. Experiments are described that allowed the authors to obtain values of the apparent mass coefficient k_{ij}, where $k_{ij} = \alpha_{ij}(\rho V_{vol})^{-1}$, and V_{vol} is the volume of the fluid displaced by the body - this case, the volume of a hemisphere. Values of k_{ij} are presented as functions of angle of attack and acceleration modulus ($\dot{V}d_P V^{-2}$). Values of k_{ij} may vary by a factor of four with acceleration modulus and the same with angle of attack. No estimations were made of practical values of acceleration moduli for dropped parachutes.

Eaton[17] noted in his paper that the complete form of the added mass tensor for a rigid axisymmetric parachute is obtained and implemented correctly for the first time. The error in Tory and Ayres is discussed as being a reversion to Henn's original faulty assumption of a real physical distinction between "included" and "apparent" mass. In Eaton's analysis, the actual masses of the canopy, rigging lines, and payload appear in the governing equations along with the four added mass components. Experiments led Eaton to run simulations with values of α_{11} to be 0.0 to 0.5 of its analogous solid-body inertial tensor component, with a baseline value of 0.2, and of α_{33} to be 0.0 to 1.0 of its analogous solid-body inertial tensor component, with a baseline value of 0.4. It is noted that the "current order-of-magnitude estimates (of α_{ij}) are very inadequate for stability studies on personnel-type, low-porosity systems."

Cockrell and Haidar[14] began the popular later approach of developing higher-order models, regarding the canopy and payload as coupled sub-systems. As for added mass effects, the authors followed Doherr and Saliaris, taking a representative value for all α_{ij}/m_a of 1.0.

Later papers continue to develop higher-order models (separate DOF for payload and canopy, ranging from 9 to 15 DOF).

Russian scientific school (Refs.18-20) proceed with the flexible structure parachute-payload and uses apparent-mass terms, obtained from the numerical simulation.

Summarizing, over a 30-year period, investigators have expressed concern over the lack of accurate dynamic modeling of apparent-mass effects, yet it appears that no studies have estimated practical values of acceleration moduli for coning, oscillating parachutes, or successfully implemented values for α_{ij} that treat them as functions of α or of acceleration module.

III.2 Simplified model

For preliminary study discussed in this paper the following simplified three-degree-of-freedom (3-DOF) model was used:

$$\dot{\bar{V}}_A = \begin{bmatrix}\dot{u}\\ \dot{v}\\ \dot{w}\end{bmatrix} = M^{-1}\left\{\frac{-qC_{D0}S_0}{V_A}\begin{bmatrix}u\\ v\\ w\end{bmatrix} + \begin{bmatrix}0\\ 0\\ mg\end{bmatrix} + \begin{bmatrix}F_x^{cont}\\ F_y^{cont}\\ 0\end{bmatrix}\right\} \quad (2)$$

where u, v, and w are the airspeed components on the axes of the body coordinate frame, α_{ii}, $i=\overline{1,3}$ are the apparent masses, m is the mass of payload and parachute assembly, q is the dynamic pressure, $S_0 = 0.25\pi d_0^2$ is a canopy surface area, g is acceleration due to gravity, and $F_{x,y}^{cont}$ are the force effect of the control actuators (in just the x and y directions). The apparent mass terms are based on the included canopy air mass m_a and are computed from the Eqns.1:

The wind speed is added to the airspeed to give groundspeed:

$$V_G = V_A + V_W \tag{3}$$

Control forces are calculated based on the pressure of the four actuators and the assumption (based on flight test data) that one control input at a time causes a 0.4 glide ratio and two control inputs at a time causes a 0.2 glide ratio. This control force is then used in the calculation of the linear accelerations of the parachute by Eqn.2, along with other parachute properties such as its mass, size, and weight, and the dynamic pressure of the atmosphere which is dependent on altitude. Linear acceleration is integrated to give airspeeds. Groundspeed is integrated to give true positions in x, y, and z coordinates of the parachute. The parachute also has a constant yaw rate ($\dot{\psi} = 0.03s^{-1}$) with small perturbations from this constant, and zero pitch and roll rates. These angular rates are integrated to give the Euler angles of the parachute, which are used to transform the coordinate axes of the parachute from the body to inertial coordinates or vice versa.

IV Derivation of the optimal control strategy

IV.1 General statement of optimization problem

Based on the AGAS concept introduced above, the optimal control problem for determination of parachute trajectories from a release point to the target point can be formulated as follows: *among all admissible trajectories that satisfy the system of differential equations, given initial and final conditions and constraints on control inputs determine the optimal trajectory that minimizes a cost function of state variables $\bar{z}$ and control inputs $\bar{u}$*

$$J = \int_{t_0}^{t_f} f_0(t, \bar{z}, \bar{u})dt \tag{4}$$

and compute the corresponding optimal control.

For the AGAS, the most suitable cost function J is the number of actuator activations. Unfortunately this cost function cannot be formulated analytically in the form given by expression (4). Therefore, we investigated other well-known integrable cost functions and used the results obtained to determine the most suitable cost function for the problem at hand.

IV.2 Application of Pontryagin's maximum principle of optimality

To determine the optimal control strategy we applied Pontryagin's principle[21] to a simplified model of parachute dynamics. This model essentially represents parachute kinematics in the horizontal plane (Figure 7):

$$\begin{aligned} \dot{x} &= u\cos\psi - v\sin\psi \\ \dot{y} &= u\sin\psi + v\cos\psi \\ \dot{\psi} &= C = const \end{aligned} \tag{5}$$

Each of four actuators in two control channels can be activated in the manner allowing the following discrete speed components in the axis of the parachute frame: $u, v \in [-V; 0; V]$. We considered these speed components as controls for the task at hand.

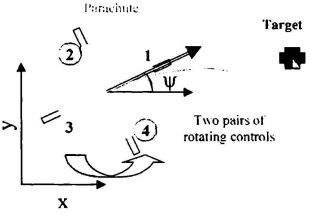


Figure 7. Projection of the optimization task onto the horizontal plane

The Hamiltonian[21] for the system (5) can be written in the following form:

$$H = (u, v)\begin{pmatrix} p_x\cos\psi + p_y\sin\psi \\ -p_x\sin\psi + p_y\cos\psi \end{pmatrix} + p_\psi C - f_0 \tag{6}$$

where equations for ajoint variables p_x, p_y, and p_ψ are given by

$$\begin{aligned} \dot{p}_x &= 0 \quad \dot{p}_y = 0 \\ \dot{p}_\psi &= (p_x, p_y)\begin{pmatrix} u\sin\psi + v\cos\psi \\ -u\cos\psi + v\sin\psi \end{pmatrix} \end{aligned} \tag{7}$$

We consider two cost functions

$$\begin{aligned} f_0 &\equiv 1 \quad \text{- minimum time} \\ f_0 &\equiv |u| + |v| \quad \text{- minimum 'fuel'} \end{aligned} \tag{8}$$

According to Ref.21, the optimal control is determined as $\bar{u}_{opt} = argmax H(\bar{p}, \bar{z}, \bar{u})$. Therefore, for the time-minimum problem the optimal control is given by

$$u = Vsign\left((p_x, p_y)\begin{pmatrix}\cos\psi \\ \sin\psi\end{pmatrix}\right),\ v = Vsign\left((p_x, p_y)\begin{pmatrix}-\sin\psi \\ \cos\psi\end{pmatrix}\right) \tag{9}$$

Figure 8 shows the graphical interpretation of these expressions. In general, the vector (p_x, p_y) defines a

direction towards the target and establishes a semi-plane perpendicular to itself that defines the nature of control actions. Specifically, if an actuator happens to lie within a certain operating angle Δ with respect to the vector (p_x, p_y) it should be activated. For a time-optimum problem since $\Delta = \pi$ two actuators will always be active. Parachute rotation determines which two. (We do not address the case of singular control, which in general is possible if the parachute is required to satisfy a final condition for heading). Figure 9 shows an example of time-optimal trajectory. It consists of several arcs and a sequence of actuations (for this example $\dot{\psi} = 0.175 s^{-1}$ and $V = 5 m/s$).

For the 'fuel'-minimum problem we obtain analogous expression for optimal control inputs:

$$\begin{aligned} p_x \cos\psi + p_y \sin\psi > V &\Rightarrow u = V \\ p_x \cos\psi + p_y \sin\psi < V &\Rightarrow u = -V \\ p_x \cos\psi + p_y \sin\psi \equiv V &\Rightarrow u = u_{s.c.} \\ -p_x \sin\psi + p_y \cos\psi > V &\Rightarrow v = V \\ -p_x \sin\psi + p_y \cos\psi < V &\Rightarrow v = -V \\ -p_x \sin\psi + p_y \cos\psi \equiv V &\Rightarrow v = v_{s.c.} \end{aligned} \qquad (10)$$

In this case actuators will be employed when appropriate dot products will be greater than some positive value. Obviously, this narrows the value of the angle Δ. In fact, for this particular cost function $\Delta \to 0$. In general any cost function other than minimum-time will require an operating angle $\Delta \le \pi$ (Figure 10).

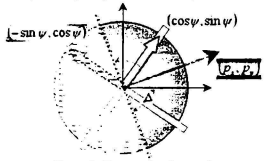


Figure 8. Time-optimal control

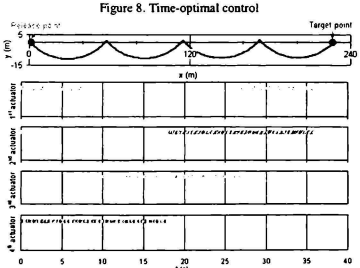


Figure 9. Example of the time-optimal trajectory and time-optimal controls

Figure 11 shows the effect of operating angle on the flight time, 'fuel' and number of actuator activations. It is clearly seen that the nature of the dependence of the number of actuations on the operating angle is the same as that of the time of flight. This implies that by solving the time minimum problem we automatically ensure a minimum number of actuations. Moreover, it is also seen that the slope of these two curves in the interval $\Delta \in [0.5\pi; \pi]$ is flat. This implies that small changes of an operation angle from its optimal value will result in negligible impact on the number of actuations. Therefore, changing the operating angle to account for the realistic actuator model will not change the number of actuations significantly.

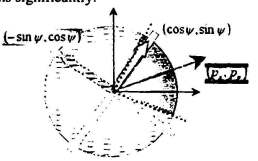


Figure 10. Generalized case of optimal control

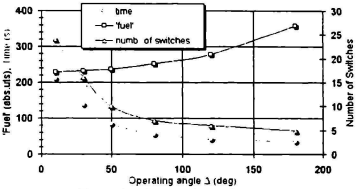


Figure 11. Influence of operating angle

IV.3 Control strategy

Preceding analysis suggested that the shape of optimal control is bang-bang. Therefore, for preliminary numerical simulation in presence of wind the control strategy was established as follows.

Considering the relatively low glide ratio demonstrated in flight test (approximately 0.4-0.5) with a descent rate of approximately $25 ft/s$[#], the AGAS could only overcome a twelve foot per second (approximately $7 kns$) wind. It is therefore imperative that the control system steers the parachute along a pre-specified trajectory obtained from most recent wind predictions. This can be done by comparing the current GPS posi-

[#] Equilibrium velocity is given by the formula $V_d = \sqrt{\dfrac{2mg}{C_{D0} S_0 \rho}}$, where ρ is a mass density of air at desired altitude.

tion of the parachute with the desired one at a given altitude to obtain the position error ($\vec{P}_r = \left(\vec{\xi}_f - \vec{\xi}_0\right)\Big|_{h=fix}$). Furthermore, to eliminate actuator 'oscillations', a tolerance cone is established around the planned trajectory (Figure 12) starting at 600*ft.* at the beginning of the trajectory and gradually decreasing to 60*ft.* at ground level. Should the position error be outside this tolerance, a control is activated to steer the system back to the planned trajectory. When the system is within 30*ft.* of the planned trajectory the control is disabled and the parachute drifts with the wind. Thirty feet was selected to encompass approximately one-sigma of the GPS errors (Selective Availability off).

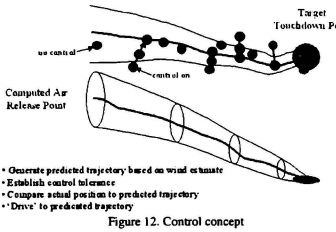


Figure 12. Control concept

As outlined above, the control system relies on the current horizontal position error to determine whether the control input is required. This position error is computed in inertial coordinate system and is then converted to the body axis using an Euler angle rotation with heading only. The resulting body-axis error (P_b) is then used to identify which control input must be activated

$$input = {}^{b}_{u}R\left(\frac{\vec{P}_e}{\left\|\vec{P}_e\right\|}\right) \qquad (11)$$

Trying to account for maximum refill time and sensors errors we chose $\Delta \approx 2.5$ instead of $\Delta = \pi$ (Figure 13). This allows the activation of a single control input or two simultaneous control inputs.

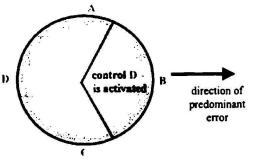


Figure 13. Control activation

Both the tolerance cone and the operating angle constraints must be active for a given PMA to be activated.

Figure 14 shows results of a simulation run that provides an insight into this control logic. The simulation uses a wind prediction profile that matches the wind profile used in the actual parachute simulation. The parachute is released at an offset from the ideal drop point of 2500*ft.* The plots show that the proper PMAs are activated (vented) when the tolerance cone and the operating angle constraints are active. One can see that at the end of the simulation that the parachute has just made it within 100*m* of the target. This brings up the concept of the "feasibility funnel." The feasibility funnel is defined as the set of points maximum distance away from the predicted trajectory for which the vehicle still has sufficient control authority to land within a certain distance from the target. The third plot in Figure 9 shows a line in the "feasibility funnel."

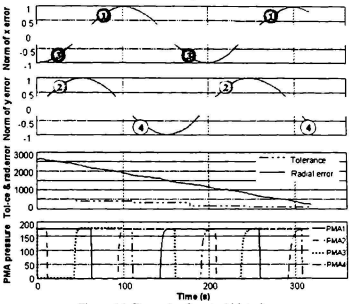


Figure 14. Example of control histories

V Simulation results

Investigation of the anticipated performance of the entire system was conducted using computer simulation incorporating the actuator model and control strategy described above. The first goal of this investigation was to determine the effectiveness of the described "trajectory-seeking" control strategy versus a control strategy that simply seeks the target landing position without using any knowledge of the winds. The second goal of this effort was to estimate the impact of changing the characteristics of the actuator system on the overall system performance.

One prerequisite to both avenues of investigation was to obtain a complete set of wind information for the drop zone. Wind information was gathered from the Yuma Proving Ground (YPG) "Tower M" drop zone using eleven Rawinsonde balloons released at one-hour intervals throughout the day on March 7th, 2000. The magnitude and direction of the wind measured by these balloons is shown in Figure 15.

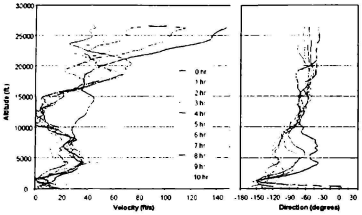


Figure 15. Measured wind velocity and direction versus altitude

In these plots, the measured wind magnitude and direction from the first balloon of the day is shown using a heavier line, and the data from the subsequent balloon releases are shown as lighter lines. Qualitative observations of wind magnitude and velocity can be gained from these plots. For the altitude range of interest, which was zero to 10,000*ft.*, the horizontal wind velocity changed by up to 20*fps* during the time span of this experiment, and wind direction in this altitude band changed by as much as 100°.

Using this wind information, a series of computer simulations of the parachute drop were conducted. The first set of simulations was run in order to visualize the effects of having a wind prediction that differs from the actual wind. The first balloon data from the day was used as a wind prediction to calculate the nominal trajectory the parachute is supposed to follow. All the subsequent balloon data were used as the actual winds, so ten actual trajectories were computed. The predicted trajectory and the ten actual trajectories are shown in a three-dimensional plot in Figure 16.

The release point of these simulated drops is directly above the origin of the horizontal plane, at an altitude of 9,500*ft*. All of the actual parachute trajectories were computed using no control of the parachute fall. The impact point (zero altitude) of the predicted trajectory has the highest positive value on the north-south axis. From the plot, it can be seen that the predicted trajectory lands north and west of the drop point, whereas the simulated drop using the last wind observation of the day lands south and west of the drop point. This observation is consistent with the wind direction data presented in Figure 16; the wind shifted from the southeast to the northeast during the course of the wind measurement experiment.

Next, a set of computer drop simulations was performed in order to assess the advantage of a control strategy that relies on a predicted trajectory based on wind information that is not current versus a control strategy that ignores the wind prediction and simply drives toward the target impact location. These two different control strategies were named "trajectory-seek" and "target-seek," respectively.

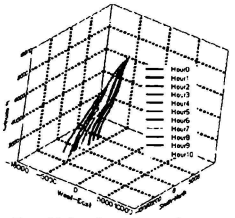


Figure 16. Sample parachute trajectories

A total of 437 simulation runs were made. On each run, the drop trajectories were computed for parachutes using both the trajectory-seek and target-seek strategies, as well as for a parachute that falls with no control inputs. Each simulation was conducted with one of the eleven Rawinsonde data files used as the actual wind for that simulation. A wind prediction was chosen at random from among the Rawinsonde data files that were gathered earlier in the day than the one selected for the actual winds. Therefore, if the data from the second balloon release of the day were chosen as the actual wind, then the data from the first balloon release of the day had to be chosen as the prediction, guaranteeing a prediction only one hour old. On the other hand, if the data from the last balloon release of the day were chosen as the actual wind, then a wind prediction from one to ten hours old could be used to compute the predicted trajectory. The predicted wind information was used in order to determine the CARP and also to determine the predicted fall trajectory for the trajectory-seek control strategy.

Another variable in these simulations was that the parachutes were dropped from a point somewhat offset from the CARP. The offset from the CARP was meant to simulate that the releasing aircraft did not hit the CARP exactly. The offsets were modeled as independent normally distributed random variables in the north-south and east-west directions. In order to determine how much of an offset to use from the CARP, an experiment was done to determine the size of the "area of attraction." This area is defined as the area around the CARP in the horizontal plane within which the parachute can be dropped and still land to within 100*m* of the target position using the onboard control system. In order to simplify this first set of experiments, the area of attraction was calculated without factoring in the effect of wind, so that the area is symmetric: a circle around the CARP in the horizontal plane. The standard deviations of the distributions of the two offsets were set at one-fourth of the radius of the area of attraction. An

overhead polar plot of the actual release and impact positions for the trajectory-seek control strategy is shown in Figure 17.

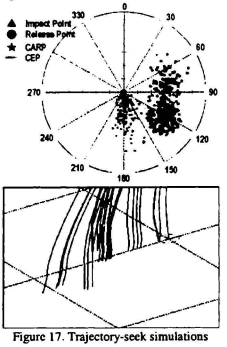


Figure 17. Trajectory-seek simulations

This plot shows that many of the release positions were southeast of the target point, relying on the predicted southeasterly winds to carry the parachute to the target point. Since the winds shifted from southeast to northeast during the day, many of the parachutes landed south of the target zone. Each of the range rings on this polar plot represents 2,000*ft*. A histogram of the simulation results is shown as Figure 18.

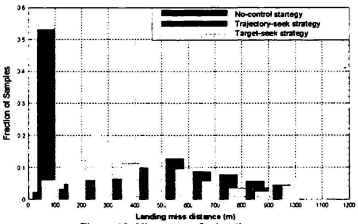


Figure 18. Histogram of miss distances

From this histogram, it can be seen that the trajectory-seek strategy showed a noticeable improvement in landing accuracy over the target-seek strategy. For the trajectory-seek strategy, over half of the simulation runs landed within 100*m* of the target, meeting the system performance goal of 100*m* of CEP. One factor affecting the data for these simulation runs was that the age of the wind predictions was not uniform. Figure 19 shows the age distribution of the wind predictions.

One can infer from this histogram that the trajectory-seek algorithm did have an advantage over the target-seek algorithm due to the fact that a substantial proportion of the simulation runs used wind data for the predicted trajectory that was fairly current.

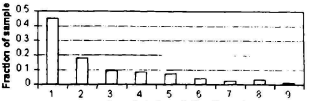


Figure 19: Distribution of wind predictions

VI Conclusions and recommendations for further research

The results presented in this paper indicate that a simple bang-bang control strategy may be sufficient to achieve the target 100*m* CEP for AGAS landing dispersion. However, further research is warranted. In particular, we propose to choose a 5-DOF model similar to the approach of White and Wolf, but with the tensor model of apparent masses and proper form of the equations from Eaton. There appears to be no current ability to resolve an accurate apparent-mass tensor as a function of anything more complex than parachute geometry. A constant yaw rate determined from flight test will be used in lieu of the 6th DOF. As a first step, the linearized 5-DOF equations will be programmed and compared to the current 2-DOF model in controlled performance. Then the nonlinear equations will be used to complete the modeling.

Aerodynamic terms will be extracted from the available terms in the literature reviewed above. Control terms will be determined from flight test data. It is of interest to estimate the relative accuracy of the current 3-DOF and the proposed 5-DOF models with regards to control strategy and required inputs. Should the added complexity of the 5-DOF model prove to contribute no apparent benefit in fidelity, the 3-DOF model will be retained.

Further analysis of the control-strategy will seek to determine the most effective operating angle Δ (because of changing aerodynamics maybe it could be better to use $\Delta \leq 0.5\pi$). Furthermore, the optimal operating angle may be asymmetric because of the different fill and vent times (Figure 21).

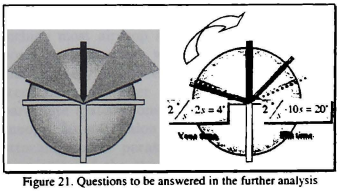


Figure 21. Questions to be answered in the further analysis

We also intend to perform additional work on determining the proper region of attraction and tolerance cone.

References

1. "Summary Report: New World Vistas, Air and Space Power for the 21st Century," United States Air Force Science Advisory Board, 1997.
2. Knacke, T.W., "Parachute Recovery Systems Design Manual," Para Publishing, Santa Barbara, CA, 1992.
3. Forehand, J.E., "The Precision Drop Dlider (PDG)," Proceedings of the Symposium on Parachute Technology and Evaluation, El Centro, CA, April 1969 (USAF report FTC-TR-64-12, pp.24-44)
4. Final Report: Development and Demonstration of a Ram-Air Parafoil Precision Guided Airdrop System, Volume 3, Draper Laboratory under Army Contract DAAK60-94-C-0041, October 1996
5. Brown, G., Haggard, R., Almassy, R., Benney, R. and Dellicker, S., "The Affordable Guided Airdrop System," AIAA 99-1742, 15th CAES/AIAA Aerodynamic Decelerator Systems Technology Conference, June 1999.
6. Dellicker, S., "Low Cost Parachute Navigation Guidance and Control," MS thesis, Naval Postgraduate School, September 1999.
7. Benney, R., Brown, G. and Stein, K., "A New Pneumatic Actuator: Its Use in Airdrop Applications," AIAA 99-1719, 15th CAES/AIAA Aerodynamic Decelerator Systems Technology Conference, June 1999.
8. White, F.M. and Wolf, D.F., "A Theory of Three-Dimensional Parachute Dynamic Stability," Journal of Aircraft, Vol.5, No.1, 1968, pp.86-92.
9. Heinrich, H.G. and Haak, E.L., "Stability and Drag of Parachutes with Varying Effective Porosity," Wright-Patterson AFB, ASD-TDR-62-100, Sept. 1962.
10. Doherr, K.-F., "Theoretical and Experimental Investigation of Parachute-Load-System Dynamic Stability," AIAA-75-1397, 1975.
11. Tory, C. and Ayres, R., "Computer Model of a Fully Deployed Parachute," Journal of Aircraft, Vol.14, No.7, 1977, pp.675-679.
12. Eaton, J.A. and Cockrell, D.J., "The Validity of the Leicester Computer Model for a Parachute with Fully-Deployed Canopy," AIAA-79-0460, 1979.
13. Doherr, K.-F. and Saliaris, C., "On the Influence of Stochastic and Acceleration Dependent Aerodynamic Forces on the Dynamic Stability of Parachutes," AIAA-81-1941, Oct. 1981.
14. Cockrell, D.J. and Haidar, N.I.A., "Influence of the Canopy-Payload Coupling on the Dynamic Stability in Pitch of a Parachute System," AIAA-93-1248, May 1993.
15. Yavus, T. and Cockrell, D.J., "Experimental Determination of Parachute Apparent Mass and Its Significance in Predicting Dynamic Stability," AIAA-81-1920, Oct. 1981.
16. Cockrell, D.J. and Doherr, K.-F., "Preliminary Consideration of Parameter Identification Analysis from Parachute Aerodynamic Flight Test Data," AIAA-81-1940, Oct. 1981.
17. Eaton, J.A., "Added Mass and the Dynamic Stability of Parachutes," Journal of Aircraft, Vol.19, No.5, 1982, pp.414-416.
18. Antonenko, A., Rysev, O., Fatyhov, F., Churkin, V. and Yurcev, Y., "Flight Dynamics of Parachute Systems," Mashinostroenie, Moscow, 1982.
19. Shevljakov, Y., Temnenko, V. and Tischenko, V., "Dynamics of Parachute Systems," Vischa Shcola, Odessa 1985.
20. Rysev, O., Belozerkovsky, S., Nisht, M. and Ponomarev, A., "Parachutes and Hang-Gliders Computer Investigations," Mashinostroenie, Moscow, 1987.
21. Pontrjagin, L., Boltjanskiy, V., Gamkrelidze, R. and Mishenko, E., "Mathematical Theory of Optimal Processes," Nayka, Moscow, 1969.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37302

AIAA-2000-4310

FLUID-STRUCTURE INTERACTION MODELING OF THE US ARMY PERSONNEL PARACHUTE SYSTEM †

Richard J. Benney, Keith R. Stein,
U.S. Army Soldier and Biological Chemical Command
Soldier Systems Center - Natick, MA 01760
Richard.Benney@natick.army.mil, Keith.Stein@natick.army.mil,

Tayfun E. Tezduyar
Mechanical Engineering and Materials Science
Rice University, TX 77005
tezduyar@rice.edu,

Michael L. Accorsi, Wenqing Zhang, and John W. Leonard
Department of Civil and Environmental Engineering
University of Connecticut, CT 06269
accorsi@engr.uconn.edu, wenqingz@engr.uconn.edu, leonard@engr.uconn.edu

ABSTRACT

We present a strategy for carrying out 3-D simulations of parachute fluid-structure interaction, and demonstrate the strategy for simulations of airdrop performance and control phenomena in terminal descent. The strategy uses a stabilized space-time formulation of the time-dependent, 3-D Navier-Stokes equations of incompressible flows for the fluid dynamics solution. A finite element formulation derived from the principle of virtual work is used for the parachute structural dynamics. Coupling of the fluid dynamics with the structural dynamics is implemented over the the fluid-structure interface, which is the parachute canopy surface. Large deformations of the structure are handled in the fluid dynamics mesh using an automatic mesh moving scheme. The strategy is demonstrated to simulate the terminal descent behavior of a standard US Army personnel parachute system. Also, preliminary simulations are presented for the response of the parachute to a variety of "riser slips," which can provide the parachute system with limited maneuverability.

INTRODUCTION

Airdrop technology is a vital Department of Defense (DoD) capability to the rapid deployment of warfighters, ammunition, equipment and supplies. In addition, airdrop of food, medical supplies, and shelters for humanitarian relief efforts are increasing in demand. A team of researchers centered at the US Army Soldier and Biological Chemical Command (SBCCOM), Natick Soldier Center (Natick) are actively involved in developing new technologies to advance DoD airdrop capabilities. An important technology being developed is the numerical modeling of parachutes and airdrop system performance utilizing high performance computing (HPC) resources. Parachutes and airdrop systems have been traditionally developed by time consuming and costly full-scale testing. The capability of using HPC tools to model and develop parachutes and airdrop systems will greatly reduce the time and cost of full-scale testing, help to minimize the life cycle costs of airdrop systems, assist in the optimization of new airdrop capabilities, and provide an airdrop virtual proving ground environment.

These modeling efforts generally involve the coupling of computational fluid dynamics (CFD) codes and structural dynamics (SD) codes. These coupled fluid-structure interaction (FSI) models are required to capture the physics of highly complex parachute dynamic phenomena. Applications and focus areas include most airdrop systems and associated phenomena. This paper will focus on the US Army's mass assault personnel parachute in terminal descent and with riser control inputs. The T-10 parachute system consists of a flat extended skirt canopy, 30 suspension lines and four 3-foot risers attaching to the paratroopers harness. A detailed description of the structural model of the canopy

†This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

and the associated fluid model for the surrounding airfield will be presented.

Paratroopers are trained at the basic airborne school at Fort Benning, Georgia. The three week course includes training in controlling a T-10 parachute (the current mass assault parachute for US Army troops) by use of riser slips. Riser slips involve physically reaching up and grabbing a riser and pulling it down to modify the canopy shape and induce glide. Riser slips are used to avoid close contact with other jumpers, avoid ground obstacles, and to "pull a two riser slip into the wind" and hold it approximately 150 to 200 feet above the ground to minimize relative horizontal ground impact speeds. Experienced paratroops can control the T-10 that is typically a non-controllable round parachute.

Riser slips are used routinely every day but are not fully understood. The effects of riser slip length, jumper weight, and other variables on the relative horizontal velocity of the system are not well known beyond jumpers practical experience. The US Army and US Air Force are exploring the use of riser slips for use in high-altitude resupply cargo systems to allow for precision landing (i.e., target accuracy) for various weight payloads. These are autonomously controlled round parachutes in which the risers are replaced by actuators and which utilize GPS as the primary navigation sensor.[1] These systems require knowledge of achievable performance in order for control optimization strategies to be developed. Currently, the only way to determine the performance characteristics is through extensive instrumented testing of full-scale systems.

In addition, the Army is exploring an Advanced Tactical Parachute System (ATPS) to replace the T-10. The canopy chosen will not use a T-10 and therefore, the systems response to a riser slip will need to be determined and measured for comparisons to the current T-10 to assess potential operational impacts. Once again, this will require extensive testing. The FSI capabilities being developed are being applied to these two real and current applications. Concurrently, and reported in this paper, the prediction of the performance of a T-10 in terminal descent and during a riser slip will be presented as a base line example. A series of detailed measurements of T-10 parachutes with personnel pulling riser slips is being conducted in partnership with the US Army Yuma Proving Ground (the primary testing agency for such systems). However, the data was not available when this paper was written.

The ultimate goal of this work is to utilize these tools to minimize feasibility testing of new systems, assist in determining the capabilities of all systems, and ultimately to reduce the life cycle cost of all DoD Airdrop Systems. This paper will provide an overview of the FSI theory and finite element strategy being developed and provide a status report on the ability to predict T-10 performance in terminal descent and during controlled riser deformations.

MODELING STRATEGY

Fluid Dynamics

For the fluid dynamics, the flow is assumed to be at a low speed and thus the Navier-Stokes equations of incompressible flows are utilized. To handle the time-variant spatial domains encountered in parachute problems, we employ the Deforming-Spatial-Domain/Stabilized Space-Time (DSD/SST) formulation[2,3] finite element formulation. In this formulation, the finite element interpolation polynomials are functions of both space and time, and the stabilized variational formulation of the problem is written over the associated space-time domain. This stabilized formulation automatically takes into account deformations in the spatial domain and protects the computation against numerical oscillations. This method has been applied to a large number of problems with moving boundaries and interfaces.

Structural Dynamics

The parachute structure is represented as a "tension structure" composed of membranes, cables, and point masses. For our problems, the structure undergoes large displacements and large rotations, but small strains. Thus, constitutive relationships are used assuming the materials are Hookean with linear-elastic properties. A finite element formulation derived from the principle of virtual work is used for the structural dynamics (SD).[4,5] Finite displacements of the structure are taken into account by using a total Lagrangian description of the problem. In addition to membrane and cable elements, a variety of parachute-specific features have been incorporated into the SD solver to include wrinkling models[6] (which eliminate compressive stresses) and time-variant cable lengths[7] (for modeling of control line pulls, "riser-slips", reefing, etc.).

Mesh-Moving Strategy

We use an automatic mesh-moving scheme to handle changes in the spatial domain due to

parachute canopy deformations. In this scheme, the fluid mesh is treated as a linearly-elastic "pseudo-solid" that deforms as dictated by the motion of the surface boundaries of the fluid domain.[8] This scheme introduces an additional computational cost associated with the mesh-moving equations, but is well suited for handling the complex geometries and arbitrary motions for this class of problems.

Fluid-Structure Coupling

The fluid-structure coupling occurs at the FSI interface, which is in this case the parachute canopy surface. We use an iterative coupling approach, with individual systems of equations being solved for the fluid and the structure. Coupling is achieved through the transfer of FSI information between the fluid and structure within an iteration loop, with multiple iterations improving the convergence of the coupled system. Displacements and displacement rates from the SD solution are treated as boundary conditions in the mesh moving scheme and CFD solver, respectively. In return, parachute surface tractions from the fluid are used as distributed forces in the SD solver. For the applications presented in this paper, we transfer only the pressure contribution from the CFD solution to the SD solver.

In the example problems, we use different meshes to represent the parachute canopy in the SD and CFD models. Biquadradic elements define the canopy in the SD mesh (to effectively represent curvatures in the membranes), whereas triangular elements define the canopy surface in the CFD mesh (in order to utilize automatic mesh generation software). Coupling information is transferred between the incompatible meshes using a least-squares projection strategy. Figure 1 depicts the incompatible meshes used for the example problems to be presented. Here, the parachute canopy is represented with 9-noded membranes elements in the SD mesh and with 3-noded triangles in the CFD mesh.

T–10 PARACHUTE APPLICATIONS

The described FSI modeling strategy is being demonstrated for a variety of parachute applications to include the the terminal descent of the Army's T–10 personnel parachute system.[9–13] The T–10 is a "flat extended skirt canopy" composed of a 35-foot diameter (D_c) canopy and 30 suspension lines each 29.4 feet long. The canopy is called a "flat extended skirt canopy" because in its constructed (or nonstressed) configuration it is composed of a main

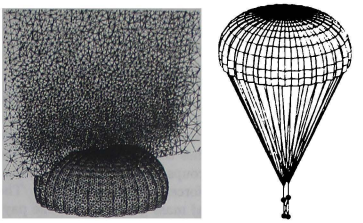


Figure 1. Incompatible meshes for T–10 parachute.

circular section with a circular vent at the apex and an inverted flat ring section, which lies under the main section and is connected to the main section at the outer radius. The lines are connected to the payload (i.e., paratrooper) with four risers. The suspension lines continue as 30 gore-to-gore reinforcements through the parachute canopy and meet at the apex. For the T–10, the vent diameter is $0.1D_c$ and the width of the skirt is also $0.1D_c$. Figure 2 shows a "blown–out" view for the SD mesh.

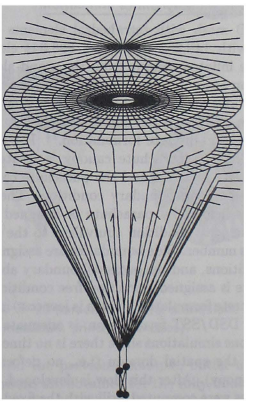


Figure 2. SD mesh for T–10 parachute.

T–10 Parachute in Terminal Descent

The following example is for the FSI of a Army personnel parachute in terminal descent freefall. The simulation process involves three main steps.

Stand-alone structural dynamics simulation:

The SD model is broken into six distinct material groups; one membrane group, three cable groups, a truss group, and a concentrated mass group. The membrane group defines the parachute canopy. We have distinct cable groups for the suspension lines, the canopy radial reinforcements, and the risers. The truss and concentrated mass groups define the payload. The SD mesh consists of 3,593 nodes, 780 nine-noded (i.e., biquadratic) membrane elements for the canopy surface, and 1,196 two-noded cable elements, and 4 concentrated mass points. The parachute system is represented by linearly elastic materials, with properties representative of a T–10. A stand-alone damped dynamic simulation is conducted for the T–10 parachute model to inflate the canopy under a prescribed differential pressure of 0.5 lb/ft^2. For the stand-alone simulation, the four payload node points are fixed in space and all other nodes are unconstrained. The equilibrium shape for the inflated parachute is depicted in Figure 1 (right). This equilibrium solution is used as the initial condition for the SD solver in the subsequent FSI simulation.

Stand-alone fluid dynamics simulation:

A 3-D mesh with tetrahedral elements is generated (with 149,253 nodes and 888,344 elements) using the inflated canopy from the stand-alone SD simulation as an interior boundary. Initial unsteady flow solutions are obtained for the fixed canopy configuration at a Reynolds number of 5×10^6 using a stabilized semi-discrete formulation.[14] For the flow simulations, the parachute canopy is treated as a zero-porosity material and the parachute canopy is assigned a no-slip boundary condition. The inflow boundary below the parachute is assigned a prescribed velocity condition equivalent to the desired Reynolds number. Side boundaries are assigned free-slip conditions, and the outflow boundary above the parachute is assigned a traction-free condition. The semi-discrete formulation, which is less cost-intensive than the DSD/SST formulation, is adequate for the stand-alone simulations since there is no time dependence in the spatial domain (i.e., no deformations of the canopy). After this flow is developed, several time steps were computed, still with the fixed canopy but by using the DSD/SST procedure, to obtain the starting CFD conditions for the FSI simulation. The developed pressure field and velocity field are shown in Figure 3.

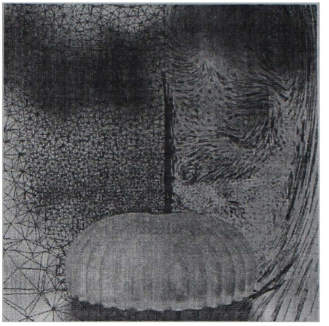


Figure 3. Inflated T–10: Initial CFD mesh and flow-field.

Stand alone FSI simulation:

Final conditions from the stand alone simulations are used to initiate the FSI simulation. Here, the parachute structure is totally unconstrained. To handle the motion of the parachute, the fluid mesh moves globally relative to the mean parachute canopy displacement and the canopy shape changes are accounted for in the CFD mesh using the automatic mesh moving scheme. The deforming parachute, with the canopy differential pressures, for the FSI simulation are shown at four instants in time in Figure 4. Figure 5 shows the computed drag force in comparison to the total gravitational force acting on the parachute system (i.e., canopy, suspension lines, risers, and payload weights). As expected, the drag force oscillates about the weight of the parachute system.

Control Line Pulls

The ability to simulate riser pulls was initially demonstrated for a "soft landing" of a T–10 FSI simulation.[13] Here, a soft-landing was accomplished by smoothly decreasing in time the natural lengths of the cable elements defining each of the four risers. The preliminary soft-landing simulation is being extended to study the response of the parachute system to single- and double-riser pulls and releases.

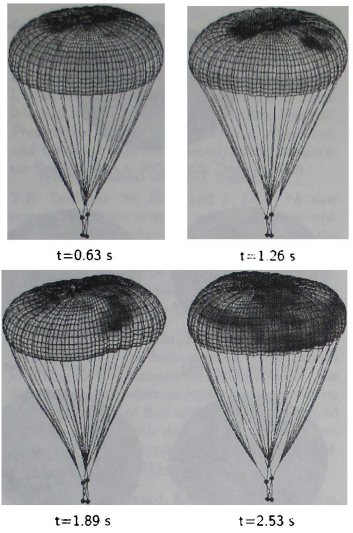


Figure 4. T–10 parachute during FSI simulation (Colored with differential pressure).

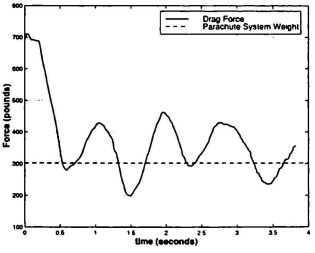


Figure 5. Drag force.

Figure 6 shows the inflated configuration for the T–10 parachute shaded with the maximum principal stresses in the canopy after stand-alone SD simulations for the standard configuration, for single- and double-riser pulls of 2.5 feet, and for single- and double-riser releases of 3.0 feet. Loss of symmetry in the parachute canopy resulting for the riser pulls and releases is evident. These asymmetries provide the canopy with lift which can be used to steer or redirect the parachute system.

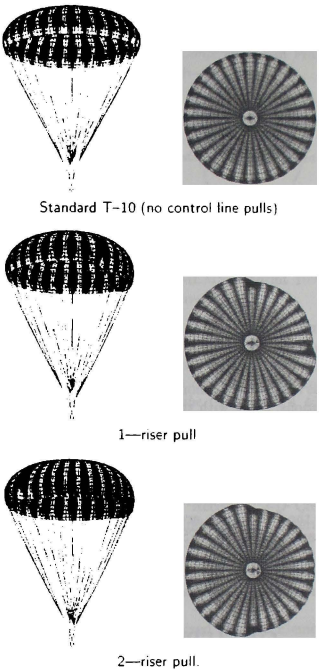
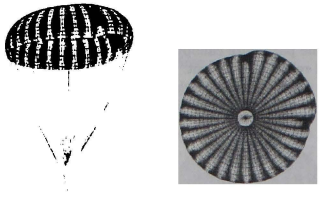
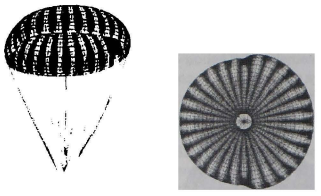


Figure 6. T–10 parachute during standalone SD simulation (maximum principal stresses).

As with the previous example, initial unsteady flow solutions are obtained for the fully-inflated configurations. Figure 7 shows the differential pressures at an instant during the stand-alone CFD simula-



1—riser release



2—riser release.

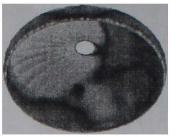
Figure 6. Continued

tions for the fixed canopy configurations. These flow solutions, along with the inflated configurations for the structure, will be used as a starting point for future FSI simulations.

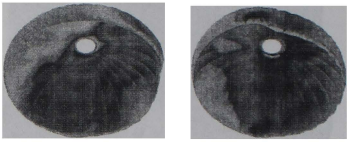
The precise shape and motions for the parachute system with and without control line pulls is governed by a strong interaction between the structure and the fluid. Thus, accurate prediction of riser-slip behavior requires a FSI model. Results to corresponding FSI simulations are not yet available. Future simulations will address the FSI behavior for these riser-slip combinations.

CONCLUDING REMARKS

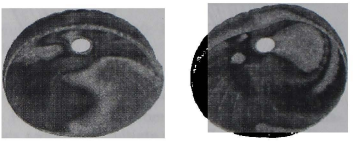
We have presented a strategy for carrying out 3-D simulations of parachute fluid-structure interaction has been presented and demonstrated for simulations of airdrop performance and control phenomena in terminal descent. The strategy is demonstrated to simulate the terminal descent behavior of a standard US Army personnel parachute system. Also, preliminary simulations are presented for the response of the parachute to a variety of "riser slips,"



Standard T–10 (no contol line pulls)



1-riser pull (2.5 feet) 2-riser pull (2.5 feet)



1-riser release (3.0 feet) 2-riser release (3.0 feet)

Figure 7. T–10 canopy during standalone CFD simulation (differential pressures).

which can provide the parachute system with limited maneuverability. Riser slips and control inputs to parachute systems (round, cruciform, parafoils, and other) are real applications for which this technology is being applied. The capability within DoD to reduce the life-cycle cost of such systems should be realized over the next several years as these modeling capabilities mature.

ACKNOWLEDGMENTS

This work was sponsored in part by AFOSR (contract number F49620–98–1–0214). The content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

1) S. Dellicker, R. Benney, S. Patel, T. Williams, C. Hewgley, O. Yakimenko, R. Howard and I. Kaminer, "Performance, Control, and Simulation of the Affordable Guided Airdrop System" *Proceedings of the AIAA Guidance, Navigaton, and Control Conference*. Aerodynamic Decelerator Session, AIAA-2000-4309, Denver, 2000.

2) T.E. Tezduyar, M. Behr and J. Liou, "A new strategy for finite element computations involving moving boundaries and interfaces – the deforming-spatial-domain/space-time procedure: I. The concept and the preliminary tests", *Computer Methods in Applied Mechanics and Engineering*, **94** (1992) 339–351.

3) T.E. Tezduyar, M. Behr, S. Mittal and J. Liou, "A new strategy for finite element computations involving moving boundaries and interfaces – the deforming-spatial-domain/space-time procedure: II. Computation of free-surface flows, two-liquid flows, and flows with drifting cylinders", *Computer Methods in Applied Mechanics and Engineering*, **94** (1992) 353–371.

4) M. Accorsi, J. Leonard, R. Benney and K. Stein, "Structural Modeling of Parachute Dynamics", *AIAA Journal*, **38** (2000) 139–146.

5) R.J. Benney, K.R. Stein, J.W. Leonard and M.L. Accorsi, "Current 3-D Structural Dynamic Finite Element Modeling Capabilities", *Proceedings of the 14th AIAA Aerodynamic Decelerator Systems Technology Conference*, San Francisco, 1997.

6) M. Accorsi, K. Lu, J. Leonard, R. Benney, and K. Stein, "Issues in Parachute Structural Modeling: Damping and Wrinkling", *Proceedings of the CEAS/AIAA 15th Aerodynamic Decelerator Systems Technology Conference*, AIAA-99-1729, Toulouse, France, 1999.

7) R. Benney, K. Stein, W. Zhang, M. Accorsi, and J. Leonard, "Controllable Airdrop Simulations Utilizing a 3-D Structural Dynamics Model", *Proceedings of the CEAS/AIAA 15th Aerodynamic Decelerator Systems Technology Conference*, AIAA-99-1727, Toulouse, France, 1999.

8) T.E. Tezduyar, M. Behr, S. Mittal, and A.A. Johnson, "Computation of Unsteady Incompressible Flows with the Stabilized Finite Element Methods–Space-Time Formulations, Iterative Strategies and Massively Parallel Implementations", *New Methods in Transient Analysis*, (eds. P. Smolinski, W.K. Liu, G. Hulbert and K. Tamma), AMD–Vol. 143, ASME, New York (1992) 7-24.

9) K.R. Stein, R.J. Benney, V. Kalro, A.A. Johnson and T.E. Tezduyar, "Parallel Computation of Parachute Fluid-Structure Interactions", *Proceedings of the 14th AIAA Aerodynamic Decelerator Systems Technology Conference*, San Francisco, 1997.

10) K. Stein, R. Benney, V. Kalro, T. Tezduyar, J. Leonard, and M. Accorsi, "Parachute Fluid-Structure Interactions: 3-D Computation", to appear in *Computer Methods in Applied Mechanics and Engineering*.

11) V. Kalro and T. Tezduyar, "A Parallel 3D Computational Method for Fluid-Structure Interactions in Parachute Systems", to appear in *Computer Methods in Applied Mechanics and Engineering*, Rice University MEMS Preprint 1999-029.

12) K. Stein, R. Benney, V. Kalro, T. Tezduyar, T. Bretl, and J. Potvin, "Fluid-Structure Interaction Simulations of a Cross Parachute: Comparisons of Numerical Predictions with Wind Tunnel Data", *Proceedings of the CEAS/AIAA 15th Aerodynamic Decelerator Systems Technology Conference*, AIAA-99-1725, Toulouse, France, 1999.

13) K. Stein, R. Benney, V. Kalro, T. Tezduyar, J. Leonard, and M. Accorsi, "3-D Computation of Parachute Fluid-Structure Interactions: Performance and Control", *Proceedings of the CEAS/AIAA 15th Aerodynamic Decelerator Systems Technology Conference*, AIAA-99-1714, Toulouse, France, 1999.

14) T.E. Tezduyar, "Stabilized Finite Element Formulations for Incompressible Flow Computations", *Advances in Applied Mechanics*, **28** (1991) 1–44.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37303

AIAA-2000-4311

UPDATED FLIGHT PERFORMANCE AND AERODYNAMICS FROM A LARGE SCALE PARAFOIL TEST PROGRAM

C. M. Madsen* and C. J. Cerimele†
NASA Johnson Space Center
Houston, Texas

Abstract

The X-38, an in-house technology demonstration program at NASA Johnson Space Center (JSC), involves a lifting body with a large scale ram air parafoil aimed at satisfying requirements for an International Space Station Crew Return Vehicle (CRV). Over the past several years of full-scale drop testing of the parafoil system, the aerodynamic database for the system as a function of angle-of-attack and flap setting has been improved and adjusted. It has been utilized throughout the program to address such issues as desired flight trim angle of attack versus rigging angle, proper brake setting position, flight maneuver design, and effects of line length ratio. This paper addresses updates and refinements to the longitudinal and lateral-directional aerodynamics, as well as recent evaluations of flight performance, of the X-38 parafoil system relative to papers by the same authors written in 1999. Several X-38 parafoil flight tests over the past two years have supplemented the original flight test base and improved the collection of steady state data for a consistent large scale parafoil system. The new tests have also covered a broader range of flight conditions and control deflections than previously reported. Although emphasis still lies on extraction of aerodynamics coefficients assuming equilibrium glide, more data has been derived on coefficient slopes and angles-of-attack to provide better simulation results of dynamic flight maneuvers. The resultant database described herein forms the basis from which design predictions for the full scale CRV parafoil (7500 ft^2) have been made. In summary, updates to the X-38 parafoil aerodynamics database, process of derivation, and application in the simulation will be presented. Hopefully, the data will provide a good general basis for understanding the parafoil flight mechanics and the development of a large-scale parafoil system.

Nomenclature

ADP	Air Data Probe, used on pallet tests
b	parafoil span
c	parafoil chord
c/4	quarter chord point on the parafoil keel
C_D	system coefficient of drag
CLD	control line deflection
C_L	system coefficient of lift
$Cm_{c/4}$	parafoil system pitching moment coefficient about c/4
L/D	lift to drag ratio
MET	Mission Elapsed Time, initiated at test article release during drop test
$\bar{q}$	dynamic pressure
RA, θ_r	parafoil rigging angle (Xpf to keel)
S	parafoil reference area
SS	stall stroke, CLD needed to stall or collapse the parafoil.
Vh	horizontal velocity
Vw	wind velocity
Vv	vertical velocity
W_{pf}	weight of the parafoil system, including rigging, but not payload
W_{sys}	weight of the parafoil system and payload
Xpf, Zpf	parafoil coordinate system: Z-axis originating at the confluence point with -Z extending up through c/4.
α	parafoil angle of attack relative to keel
ϕ_w	direction of the wind measured from the north (i.e., wind is coming *from* ϕ_W).
γ	flight path angle
θ	parafoil pitch angle (Xpf to horizon)
θ_t	total parafoil pitch angle (keel to horizon)
ρ	atmospheric density

* Aerospace Engineer, Aeroscience and Flight Mechanics Division.
† Deputy Branch Chief, Aeroscience and Flight Mechanics Division, Member AIAA.
Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental Purposes. All other rights are reserved by the copyright owner.

Introduction

NASA Johnson Space Center is currently developing an in-house technology demonstration and prototype of a crew return vehicle for the International Space Station called the X-38. Use of a large scale parafoil for the final flight phase of earth atmospheric re-entry is intended to provide a safe recovery on land. Development of the parafoil system originated with research into wind tunnel tests, previous flight test programs, and subscale drop testing and tow testing. Eventual large scale testing of a parafoil system led to understanding of some of the flight mechanics and development of an initial aerodynamics database as described in three papers[1,2,3] published in 1999. History of the large scale (5468 ft^2) drop tests to date are shown in Table 1.

Test ATI #	Test Date	Wing (ft2)	θ riggin	W/S (psf)
P2D13	13-May-97	5468	13°	2.43
P2D14	14-May-97	5850	13°	3.52
P2D15	11-Jul-97	5468	13°	2.45
P2D16	12-Jul-97	5468	13°	3.84
P2D17	8-Sep-97	5468	13°	2.51
P2D19	8-Oct-97	5468	13°	2.60
P2D20	12-Dec-97	5468	13°	2.52
P3D1	1-Apr-98	5468	13°	2.52
P2D21	4-Jun-98	5468	13°	2.58
P2D22	8-Oct-98	5468	13°	2.60
P2D23	8-Oct-98	5468	10°	2.57
P2D24	4-Nov-98	5468	13°	2.68
P2D25	15-Dec-98	5468	10°	2.50
P2D26	16-Dec-98	5468	10°	2.65
P3D2	6-Feb-99	5468	10°	2.34
P3D3	5-Mar-99	5468	10°	2.47
P3D4	9-Jul-99	5468	10°	2.64
P2D27	28-Oct-99	5468	10°	2.57
P3D5	30-Mar-00	5468	10°	2.65

Table 1: X-38 Drop Tests with 5468 ft^2 Parafoil

The historical wind tunnel test data[4, 5, 6] and the X-38 program drop tests up through P2D26 provided the database upon which the original AIAA papers pertaining to the X-38 parafoil development were built. These original papers presented a trend analysis of the longitudinal coefficients in order to describe the performance of the parafoil as a function of flap setting and in the presence of conditions such as "reflex". Merging all the sets of longitudinal aerodynamics required assuming values for steady state angles of attack. This assumption was based on early flights and subscale testing. New techniques of measuring angle-of-attack were alluded to at the time, which have now been successfully employed.

In addition, the parafoil rigging angle (the angle between the parafoil's X-body axis and the bottom surface of the canopy) has been changed from the previous reports. The initial reports were based on testing primarily of a 13 degree rigging angle while this paper focuses on the updated rigging angle of 10 degrees. The change was made to provide improved flight performance resulting from trim at a higher angle-of-attack than initially flown. Coupling the previous data with new test data, primarily from drop tests of two 5468 ft^2 parafoils, has provided a better overall database of trends with respect to angle-of-attack and flap settings. Future small scale tow testing and flight testing with the improved sensor systems can provide a much larger range of angle-of-attack data from changes in rigging angle than allowed in full scale flight tests. Due to the fact that this data is not yet available, the results presented herein still are limited to a relatively small range of angle-of-attack and contain some amount of uncertainty. From the resultant database, however, more accurate analysis of response to gusts and control inputs, stability margins, and sensitivities to rigging angles can eventually be made. The original corridor of rigging angle versus trim angle-of-attack has also been updated.

Knowledge of the lateral-directional aerodynamics and quantifying the turn performance of the large-scale parafoil system has also improved. Flight tests have included a variety of turn maneuvers investigating the capabilities of large-scale parafoils to accomplish objectives such as turning for range, energy management and spiral recovery. The data collected during these tests have been analyzed to characterize the parafoil's turn performance as a function of flap deflection. The range of flap deflection deltas has been expanded to provide better definition of the trends of the directional aerodynamic coefficients. Comparison of the performance of the 5468 ft^2 wing with the two major rigging angles that have been utilized in his program will be shown. In addition, refinement of the adverse turns due to "reflex" will be presented.

The new aerodynamic database subsequently has been used to update the eight degree of freedom (8 DOF) simulation that models the parafoil and payload in flight. The simulation has been used to successfully reconstruct multiple flight tests, and accuracy of reconstructed parameters are presented. Specifically, the goal was to attempt to define one aerodynamic database that reconstructs well the flights of both primary tested rigging angles. Dynamic maneuvers such as flares and brake release surges, as well as steady state performance, are used for comparison to flight data. In order to evaluate stability margins, the simulation can be stressed to evaluate the extremes in angles-of-attack and how close they approach what is believed to be the safe flight limits. In the future, with additional subscale testing of a larger range of angles-of-attack, an expanded aerodynamic database may allow confident simulations which show the "stall" of the parafoil or re-trim to an adverse angle-of-attack.

Parafoil Models

The majority of the data presented herein was derived from recent tests of two parafoil wings of area 5468 ft^2. The aspect ratio was approximately 2.7 and line length to span ratio was 0.6. Basically, the wing has a Clark-Y airfoil shape with a 43 degree inlet angle. The initial rigging angle, θ_r (Figure 1), as in the previous papers, was 13 degrees, later being changed to 10 degrees for performance improvements.

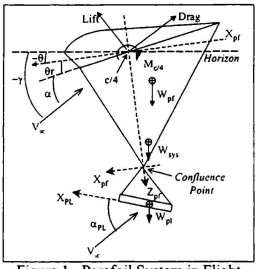


Figure 1: Parafoil System in Flight

Flaps deflections are exercised on only the outer ¼ span of each wingtip. The total "flyable" stroke of the control lines or deflection of the flaps was termed "stall stroke" and was empirically derived by tow testing and flight testing, some on small scale parafoils. "Stall stroke" was defined as the flaps deflection required to induce a rearward folding of the wingtips brought on by flap drag and structural deformation of the canopy. Note that the stall stroke does not necessarily imply a stall in the typical sense of a wing exceeding the stall angle-of-attack. It is simply the stroke which creates a span-wise structural collapse. The stall stroke changes as a function of the rigging angle due to the change in wing incidence to the flow. As the rigging angle is decreased, producing larger angle-of-attack, the stall stroke becomes less due to the fact that the flaps become more effective for a given deflection. For the 13 degree rigging angle, the stall stroke was defined as 314 inches, while for the 10 degree rigging angle it was 227 inches.

The stall stroke is sometimes used herein as a non-dimensionalizing parameter for flaps deflection, ie. % stall stroke or %SS. It must be noted that whenever the non-dimensional deflection is quoted, it actually represents different physical stroke lengths depending on the rigging angle of the parafoil being referenced. Therefore, the airfoil geometry is actually different between the 10 degree and 13 degree rigging angle configurations when quoted at the same %SS, other than 0%. When stroke is referenced in inches or feet however, the airfoil geometry is ideally the same between the two rigging angle configurations. Herein, the flap deflection cited in actual length shall be referred to as absolute flap deflection, whereas the stroke as a percent of stall stroke shall be referred to as relative flap deflection.

Longitudinal Aerodynamics Model

Performance Parameters

As was the case in the previously reported studies, measurement of dynamic pressure and angle-of-attack was both difficult and critical to the correct derivation of a longitudinal aerodynamic database for the parafoil. Dynamic pressure was found to be most reliable from the flush air data system (FADS) on the vehicle drop tests and from the air data probe (ADP) on the pallet drop tests. In all cases, the measurement of winds (from balloons) and ground relative velocities (tracking or GPS) was critical for approximate validation of the measured values. In some cases, calibration of the source data was performed by comparison to relative velocities derived from the balloon winds and tracking or GPS. Relative flight path angle was generally derived from altitude rate from the tracking or GPS in concert with a relative horizontal velocity derived from the measured dynamic pressure.

Relative horizontal velocities, vertical velocities, and flight path angles in trimmed flight were compiled for

the two different rigging angle parafoils over the series of flights shown in Table 1. The data generally laid good trends of these parameters as a function of %SS when scaled to identical wing loading at sea level as seen in Figures 2, 3 and 4. Wing loading for these plots is about 2.6 psf. In general, the higher velocities and steeper flight paths of the larger rigging angle are indicative of a smaller trim angle-of-attack.

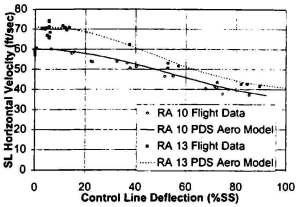


Figure 2: Trim Relative Horizontal Velocity Data for Two Rigging Angles

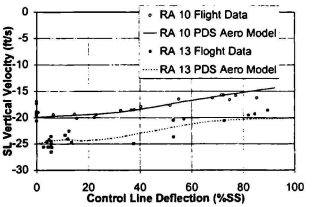


Figure 3: Trim Vertical Velocity Data for Two Rigging Angles

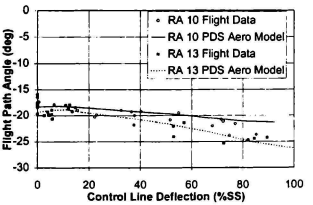


Figure 4: Trim Relative Flight Path Angle Data for Two Rigging Angles

Lift and Drag

Longitudinal trim aerodynamic coefficients of lift and drag derived from the performance of the parafoils also presented good curves as a function of the relative flaps deflection. These are shown in Figures 5, 6, and 7. Again, the coefficients and lift-to-drag ratio suggest a lower angle-of-attack for the 13 degree rigging angle than the 10 degree. Drag coefficients and, to a lesser extent, lift coefficients show the "reflex" effect described in reference 1 in the region of small flap deflections. It appears to be more evident for the 13 degree rigging angle than the 10 degree. It is manifested by the relatively small change in drag as the flaps are initially pulled down. In fact, the curves are fit to the 13 degree RA data with an initial decrease in drag with flap deflection.

In effect, this is the reflex being "pulled out" of the canopy. A "dead band" of flap effectiveness is essentially created for the first 10-15% of SS. Because of this phenomenon, the turn performance of the canopy exhibits a turn reversal for flaps of 0% on one side and small deflections on the other.

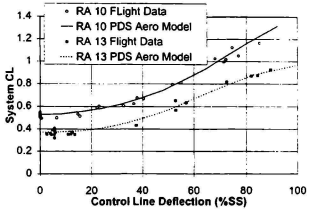


Figure 5: Derived Lift Coefficient Data for Two Rigging Angles

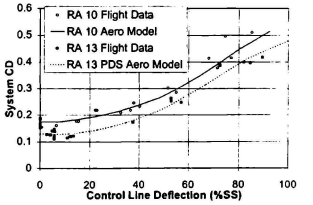


Figure 6: Derived Drag Coefficient Data for Two Rigging Angles

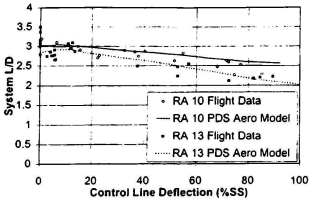


Figure 7: Derived Lift-to-Drag Ratio Data for Two Rigging Angles

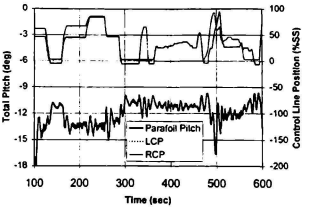


Figure 8: Example of Pitch Data Derived from Parafoil Inclinometer System

The key to development of a good aerodynamic database from the trim coefficients is to relate them to angle-of-attack in a manner that produces the same database for both rigging angles. This must be the case since the parafoil shape remains the same despite changes in rigging angle for a given absolute flap deflection, except for perhaps in the region of reflex effect. The original estimate for trim angle-of-attack, as cited in reference 1, for the 13 degree rigging angle was 4 degrees with zero flaps. This number was based in part on photographs and video. This value of 4 degrees trim alpha was also assumed to stay nearly constant for all other flap deflections. Angle-of-attack measurement methodology, however, was significantly improved recently over the previously reported studies. Unfortunately, the new method has only been applied to the 10 degree rigging angle flights and thus, the 4 degree estimates of trim alpha still exist for the 13 degree rigging angle.

The new advancement in angle-of-attack measurement was achieved by implementation of an inclinometer system mounted in the parafoil. This system involves a set of eight single axis accelerometers mounted along the centerline of the parafoil to measure the gravity vector in a steady state flight situation. To function properly, fairly long segments of flight with constant flaps was required to get good pitch angle data from the inclinometer system, typically on the order of 30 seconds. Even with the sufficiently long period of constant flaps, heavy filtration of the inclinometer system data was required. An example of the filtered pitch data is presented in Figure 8.

Based on the inclinometer data, estimates of trim angle-of-attack versus relative flap deflections for the 10 degree rigging angle configuration are shown in Figure 9. Compared to the assumed constant trim value of the 13 degree RA, the trim alpha varies from approximately 6.5 degrees for zero flaps to 8.5 degrees for 90% SS with the 10 degree RA. Although this raises some question as to the constant assumption of trim alpha for the 13 degree RA, it will be seen that the resulting slopes of aerodynamic coefficients as a function of alpha in the derived database appear reasonable. Also the resulting database predicts a range of flyable rigging angles that agrees well with the range witnessed in subscale drop tests. This will also be shown more clearly once the full aerodynamic model is presented.

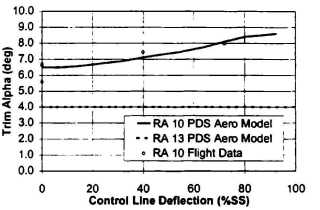


Figure 9: Trim Angle-of-Attack Data and Estimated Curvefits for Two Rigging Angles

Note that indeed the trim angle-of-attack increases with decreasing rigging angle. The mechanics behind this process are as follows. As the rigging angle is changed, a new trim pitch attitude must result that causes the moments due to weight of the payload and canopy to offset aerodynamic moments at a new angle-of-attack. At the same time, the L/D generated by the new angle-of-attack produces a flight path that combines with the new pitch angle resulting in the same angle-of-attack.

With the aerodynamic coefficients of lift and drag defined at two points of angle-of-attack resulting from the two rigging angles, curves of lift and drag versus

alpha were generated for the various flap settings (Figures 10 and 11). The basic shapes of these curves depended somewhat on the previous reports and wind tunnel tests, but have more sound basis in flight derived data now. Some differences are noted from the previously reported database. Reflex effects, and their difference with rigging angles, tends to change the slopes of the coefficients at the small flap deflections and has caused curves to cross at very low alphas. Peak lift coefficients also had to be increased and moved out to larger angles-of-attack than previously reported due to the new inclinometer measurements with the 10 degree rigging angle. Zero lift alpha has decreased a few degrees. In general, however, the curves have maintained the expected shapes for a typical wing.

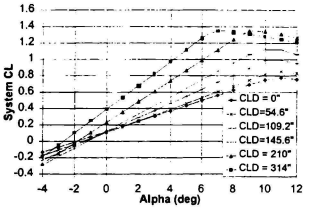


Figure 10: Current 5468 ft^2 Parafoil Lift Coefficient Database

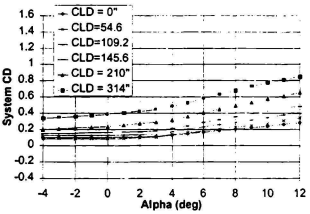


Figure 11: Current 5468 ft^2 Parafoil Drag Coefficient Database

Pitching Moment

Perhaps the most significant change to the old database concerns the pitching moment curves as a function of angle-of-attack. These curves are a direct result of forcing two trim angles-of-attack (corresponding to the two rigging angles) to agree with Figure 9 at specific flap settings. Using the flight derived lift and drag coefficients as in Figures 5 and 6, pitching moment coefficients about the quarter chord were calculated to produce the desired trim alpha as described in reference 1. The previous report only utilized one trim angle per flap setting and arbitrarily maintained similar Cm-alpha slopes for each flap setting. Now with two trim angles available for each flap setting, the Cm-alpha slopes are seen to change with flaps, as shown in Figure 12. Basically the curves are linearized between the two known trim alphas and smoothed into previous curves at the extremes in angle-of-attack.

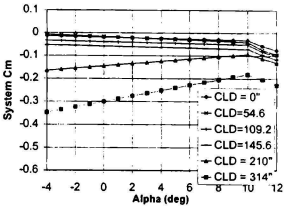


Figure 12: Current 5468 ft^2 Parafoil Pitching Moment Coefficient Database

These pitching moment slopes are obviously directly dependent on the trim alphas associated with each flap setting, as are the slopes of the lift and drag coefficients. It was thought that varying these trim angles and associated coefficient slopes in a simulation could lead to confidence in the selected trim points by comparing dynamic responses to flight data. However, the responses were too invariant with these changes to produce such a validation.

Implications of the Longitudinal Model

To summarize the current longitudinal aerodynamic model, Figures 10, 11, and 12 represent flight derived aerodynamics which generate the prescribed trim angles-of-attack produced by two rigging angles. The prescribed trim angles-of-attack are still somewhat questionable, particularly for the 13 degree rigging angle. This is hoped to be verified with additional sub-scale and full-scale testing and usage of the parafoil inclinometer system.

With the new aerodynamics model, effects of rigging angle changes can be predicted. Figures 13, 14, and 15 show the influence of rigging angle on the trim angle-of-attack, total pitch angle, and flight path angle for zero flap deflection. The sensitivity of trim alpha to rigging angle appears to be about 0.9 deg/deg. If angle-of-attack limits are assumed to be +12 degrees to 0

degrees based on previous research, then the flyable range of rigging angles would be predicted to be approximately 5.5 degrees to 20 degrees.

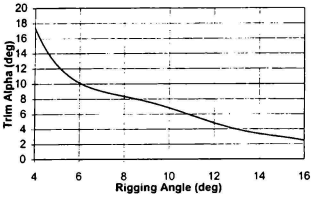


Figure 13: Trim Angle-of-Attack Vs. Rigging Angle Derived from Current 5468 ft² Parafoil Database

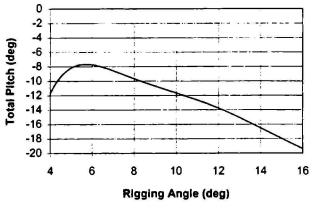


Figure 14: Trim Total Pitch Angle Vs. Rigging Angle Derived from Current 5468 ft² Parafoil Database

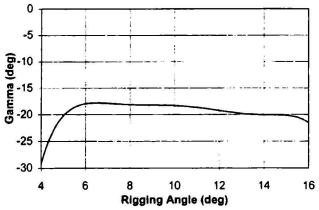


Figure 15: Trim Flight Path Angle Vs. Rigging Angle Derived from Current 5468 ft² Parafoil Database

The trim alpha curve shown in Figure 13 produces trim aerodynamics as shown in Figures 16, 17, and 18. Maximum L/D is generated by a rigging angle of approximately 7 degrees corresponding to a trim angle-of-attack of 9.5 degrees for zero control stroke.

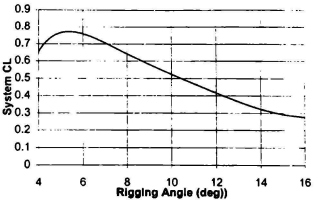


Figure 16: Trim Lift Coefficient Vs. Rigging Angle Derived from Current 5468 ft² Parafoil Database

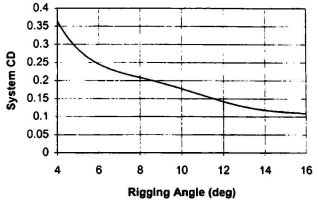


Figure 17: Trim Drag Coefficient Vs. Rigging Angle Derived from Current 5468 ft² Parafoil Database

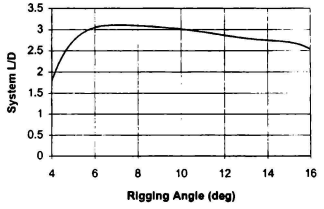


Figure 18: Trim Lift-to-Drag Ratio Vs. Rigging Angle Derived from Current 5468 ft² Parafoil Database

Directional Aerodynamics Model

Turn Performance

As was suggested in an earlier paper[2], steady state turn rate performance appeared to be greater for smaller rigging angles for a given absolute base and delta flap deflection. This postulate still seems to hold true in comparing the turn rates of the 10 degree RA to the 13 degree RA, as in Figure 19. Perhaps this seems to be

intuitive in that the flaps are more effective at the higher trim angle-of-attack of the 10 degree RA. The effects of reflex are witnessed in the turn reversals for 0% base deflection and small delta deflections. As in the longitudinal aerodynamics, the reflex appears to effect the higher RA (lower trim alpha) more than the lower RA (higher trim alpha).

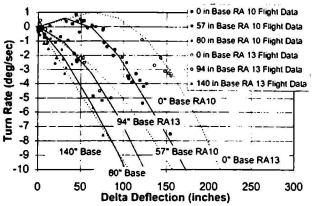


Figure 19: Turn Rates Data Vs. Absolute Flaps Deflection for Two Rigging Angles

Interestingly, it was discovered that when these same turn rates are plotted as a function of relative flap deflection (%SS), the performance of the two rigging angles appears to be nearly equal (Figure 20). This observation has not yet been determined to be purely coincidental or a result of the physical equations.

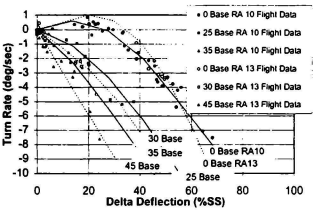


Figure 20: Turn Rates Data Vs. %SS Flaps Deflection for Two Rigging Angles

Yawing Moment

Yawing moment coefficients due to delta flap deflections were derived for the parafoil with the same approach used in reference 2.

$$\delta_f Cn_{\delta_f} = \frac{(turn\ rate)\ Cn_r b}{2Vtot}$$

It must be noted that the assumption of constant yaw damping derivative was still employed. In fact, the same yaw damping derivative was used for both rigging angles. Figures 21 and 22 show these yawing moment coefficients as a function of base and delta flap deflection. Note that the lower rigging angle appears to produce more yawing moment than the higher rigging angle. Again this may seem to be intuitive due to the higher angle-of-attack of the lower RA.

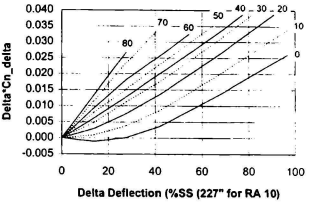


Figure 21: Current 5468 ft² Parafoil Yaw Moment Database for Rigging Angle of 10 Degrees

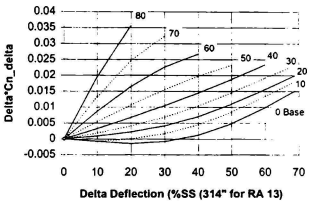


Figure 22: Current 5468 ft² Parafoil Yaw Moment Database for Rigging Angle of 13 Degrees

The equal turn rate performance of the two RA parafoils when plotted versus relative flap deflections (%SS) can be explained in the following manner. Because the lower rigging angle flies at lower speed, the yaw damping contribution to the yaw moment coefficient is greater than that of the higher rigging angle (seen in the equation above). However, the moment coefficient due to delta deflection is greater for the lower rigging angle due to higher trim alpha. When the base and delta deflection is considered in the absolute sense, the larger yaw moment generated by the lower RA outweighs the larger damping. That is, the ratio of flap contribution to damping contribution is larger for the smaller RA. However, when the

deflections are considered in %SS (effectively giving more base and delta deflection to the higher rigging angle), the ratio of the flap moment to the damping apparently remains about the same for each RA. This results in similar steady state turn rates.

Implications of the Directional Model

The directional model represented by Figures 20, 21, and 22 has an interesting potential implication. If indeed the turn rate model does not change with rigging angle when viewed versus flap deflections in %SS, then perhaps measured turn rates for an arbitrary rigging angle with known absolute flap deflections can allude to the correct stall stroke. In addition, by comparison to the stall strokes of the 10 and 13 degree RA's, an inference of rigging angle and trim angle-of-attack at zero flaps could potentially be made.

Trajectory Reconstructions

Free Flight

The true test of the adequacy of the derived aerodynamic database is the ability to accurately reconstruct actual parafoil flights in a simulation. The same simulation was utilized as in the last reports - the 8-degree-of-freedom Parafoil Dynamics Simulation (PDS). One of the problems noted in the previous studies was the inability to match some dynamics in velocities. It has been discovered that, in large part, this was due to the usage of wind corrected flight data and comparing to simulations with no winds. The effects of changing wind directions or magnitudes are not adequately represented by the simulation without winds. For example, if in steady flight a head wind is increased, the vertical velocity typically reduces in magnitude while the air relative horizontal velocity increases temporarily until equilibrium is recovered. The wind corrected flight data (air relative) would show these variations in velocities. The simulation with no winds would of course show a constant horizontal and vertical velocity. Adding flight-derived winds to the simulation improved the replication of dynamics. These winds were calculated using the payload air data system, inertial tracking, and attitude systems. Figure 23 shows the air data derived winds versus the wind balloon data collected during P3D5. The trends are similar, but the air data derived wind captures the higher frequency gusts and wind variations much better. Figure 24 shows a PDS reconstruction (with winds input) of vertical velocity from flight test P3D5 compared to inertial flight data. The match is quite good – within approximately 10 % on average.

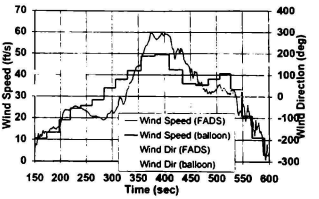


Figure 23: Flight Air Data System Wind Extraction for P3D5

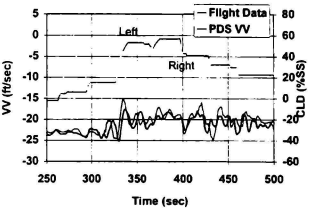


Figure 24: PDS Reconstruction of Vertical Velocity from Drop Test P3D5

Unfortunately, introduction of earth-relative winds into the simulation leads to horizontal velocity errors when turn rates aren't exactly matched. If a turn in the simulation does not exactly match that of flight, the resulting heading is not the same relative to the wind as it was in flight. This leads to ground relative horizontal velocity errors. Therefore, straight flight is well reconstructed by the simulation with winds (as in Figure 25 for P3D5) whereas long flights with turns

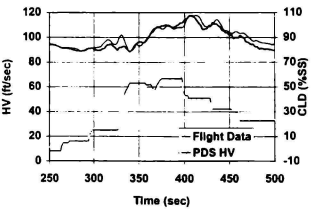


Figure 25: PDS Reconstruction of Horizontal Velocity from Drop Test P3D5

eventually are not well reconstructed in horizontal velocity. To show horizontal velocity reconstructions of long flights with turns, it is best to show the simulation without winds compared to wind corrected data and simply note that some dynamic errors will be common. Accuracy is typically within about 5 %.

Turn rates, as measured by vehicle yaw rates, are not effected by winds and thus re-create fairly well as seen in Figure 26 for P2D27. Accuracy appears to be within 1 degree/sec.

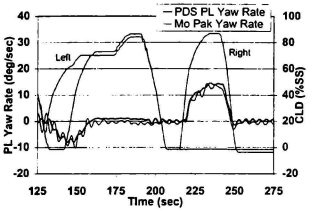


Figure 26: PDS Reconstruction of Yaw Rate from Drop Test P2D27

Reconstruction of a flare maneuver is shown in Figures 27, 28, and 29 for vertical velocity, horizontal velocity, and pitch rate, respectively. Accuracy appears to be within 10% in these velocities. A look at the simulated angle-of-attack during the maneuver show a peak of about 8.5 degrees (Figure 30). Stall angle-of-attack was previously estimated as about 12 degrees, which implies a 3.5 degree margin during the flare maneuver.

Hopefully, the same aerodynamics database would reconstruct a 13 degree RA flight as well as the 10 degree RA examples shown above. Previous papers

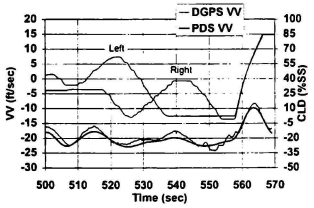


Figure 26: PDS Reconstruction of Vertical Velocity in a Flare from Drop Test P2D27

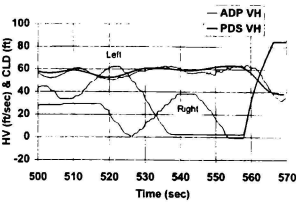


Figure 28: PDS Reconstruction of Horizontal Velocity in a Flare from Drop Test P2D27

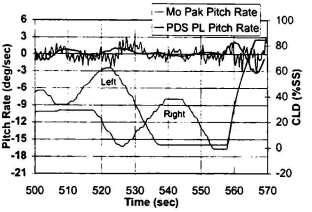


Figure 29: PDS Reconstruction of Pitch Rate in a Flare from Drop Test P2D27

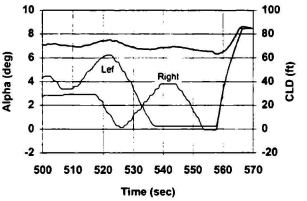


Figure 30: PDS Reconstruction of Angle-of-Attack in a Flare from Drop Test P2D27

have shown good agreement for the 13 degree RA, although with the changes introduced to the database, these cases still need to be re-analyzed to ascertain adequate reconstructions.

Conclusions

An updated X-38 large scale parafoil system aerodynamics database has been developed based on

flight tests. Changes to previous reports on this subject include improved knowledge of trim angles-of-attack, better supported slopes of aerodynamic coefficients, updated effects of rigging angle, and improved understanding of turn rates. One database of longitudinal aerodynamics as a function of alpha and flaps settings has been shown to adequately represent two different rigging angles. The database approximately predicts the range of flyable rigging angles seen in subscale drop tests, as well as generates true trim flight conditions of the large scale parafoil. New correlations of turn rates to flap deflections have also been defined. Use of the updated aerodynamics database in parafoil flight simulations of a 10 degree RA parafoil can closely match actual flight velocities, attitudes, and attitude rates. Flares and dynamic events are reconstructed within good approximation, as well as steady state flight.

Future drop tests and tow tests are intended to further verify the slopes of coefficients to even larger ranges of angle-of-attack through expanding the range of rigging angles. Additional simulated and analytical studies will hopefully produce improved understanding of the dynamic parafoil modes and identify margins in such situations as gusts, flares, spirals, and brake releases.

Acknowledgements

The authors wish to thank co-workers Jenny Stein and Koki Machin for leadership of the X-38 parafoil test program; Pioneer Aerospace and Sandia National Labs for continued support and development of drop tests; and Paul Royal and Dwayne Smith of Lockheed Martin for instrumentation assistance.

References

[1] Iacomini, C. Schroeder and Cerimele, C., "Longitudinal Aerodynamics From A Large Scale Parafoil Test Program", AIAA-99-1732, (1999)

[2] Iacomini, C. Schroeder and Cerimele, C., "Lateral-Directional Aerodynamics From A Large Scale Parafoil Test Program", AIAA-99-1731, (1999)

[3] Iacomini, C. Schroeder and Madsen, C., "Investigation of Large Scale Parafoil Rigging Angles: Analytical and Drop Test Results", AIAA-99-1752, (1999)

[4] Geiger, Robert H. and Golden, Ronald A., "Pioneer Aerospace Corporation, Advanced Recovery Systems Wind Tunnel Test Report", Series 2, Volume 1, ARS-WT-2, August 1990

[5] Ware, George M. and Hassell, James L., "Wind Tunnel Investigations of Ram-Air-Inflated All-Flexible Wings of Aspect Ratios 1.0 to 3.0", NASA TM SX-1923, December 1969

[6] Nicolaides, John D., "Parafoil Wind Tunnel Tests", AFFDL-TR-70-146, Notre Dame University, June 1971

[7] Anderson, J. D. Jr., Fundamentals Of Aerodynamics, Second Edition, McGraw-Hill, Inc., New York, 1984

[8] Lingard, J.S., Gliding Parachutes, Parachutes Systems Technology Short Course, Minneapolis, Minnesota, October, 1998.

[9] Lingard, J. S., "Ram-Air Parachute Design", 2nd ADS Technology Seminar: Precision Aerial Delivery, Clearwater Beach, Florida, May 1995

[10] Martin, F., "Parafoil Aerodynamic Chracteristics Derived from Flight Measured Suspension Loads", AIAA-99-1734, (1999)

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37304

AIAA-2000-4312

DESIGN AND TESTING OF THE X-38 SPACECRAFT PRIMARY PARAFOIL*

Ricardo A Machin
X-38 Parachute Test Director
NASA Johnson Space Center

Abstract

In August of 1995 the National Aeronautics and Space Administration (NASA) began testing large ram-air parafoils for potential use in the landing phase of spacecraft recovery. This effort eventually became a part of the X-38 project, a technology demonstrator for the International Space Station Emergency Crew Return Vehicle.[1] This paper traces how the original parafoil, used in the U. S. Army Guided Precision Air Delivery System (GPADS), was modified to improve the initial deployment dynamics. A discussion of how test experience with 750 ft^2 parafoils has been used to scale up to the fullscale 5500 ft^2 and 7500 ft^2 parafoils is presented. In particular, the development of two techniques that have greatly improved the repeatability of the parafoil initial deployment and decreased the first stage deployment dynamics are discussed: a first stage upper surface energy modulator and floor vents. Comparisons of trends and results from the various scale parachutes is presented as well as constraints that at times have driven the direction that the design has taken.

Introduction

One of the principle challenges during the development of the parafoil system for the X-38 program has been to achieve a repeatable, low dynamics on heading opening of the fullscale ram-air parafoil. The X-38 parafoil is deployed by the primary drogue, only after the drogue has achieved a steady state dynamic pressure and the system is at a flight path angle of 90 degrees. The decision to use the drogue to deploy the parafoil was made in order to avoid the additional time delay associated with deploying a pilot parachute which in turn would deploy the parafoil. The increase in dynamic pressure over this additional time in free-fall would require a decrease in the size of the parafoil first stage, with the drawbacks of requiring an additional stage of reefing overall for the parafoil and accepting worse initial inflation dynamics.

Based on experience from GPADS and early X-38 fullscale testing, upon reaching line stretch the canopy would rebound back towards the payload, with the inertial forces from rebound overcoming the initial drag area of the first stage canopy. Eventually the canopy would begin to expand into the first stage planform. The canopy drag area would overcome the rebound from line stretch, restoring tension to the suspension lines and start decelerating the payload. This is the major drawback of using the drogue to deploy the parafoil, it is deployed at very high line stretch velocities.

Once the parafoil spreads to the rectangular first stage planform, it surges forward from a ballistic trajectory to establish the forward flight associated with a parafoil. Usually the surge was more like a charge with the leading edge partially collapsing during this transition to forward flight. The chordwise shape of the parafoil would be crushed shorter momentarily before once again achieving the intended first stage planform shape after reaching the peak point the forward surge. The primary reason that the leading edge is overrun during this surge is that the interior cells of the canopy are not fully pressurized at this point in the inflation process. The canopy can only pressurize with air that enters at the leading edge inlet.

What occurred next depended on the magnitude of the surge. If the surge was large enough, the parafoil would get well off the wind line and the payload would swing through catching up to and passing the parafoil. This would set up a two body motion that would not damp out until well into the later stages of the parafoil disreefing. While this motion would almost always settle out before releasing the parafoil deployment brakes, the randomness with which the openings were occurring was resulting in very dynamic early stages. Often time full revolutions of twist would develop between

* Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental Purposes. All other rights are reserved by the copyright owner.

the parafoil heading and the forward direction of the vehicle by the time that the parafoil was completely disreefed.

With a lifting body beneath the parafoil, the lifting body having large fins aft of the center of gravity, the possibility exists of not untwisting completely once in forward flight under the parafoil. Twist between the lifting body and the parafoil could compromise the flight characteristics of the system and cause the winches to be less effective in deflecting the control surfaces of the parafoil. The dynamic openings were also considered unacceptable from a crew comfort point of view. Therefore controlling the transition of the first stage parafoil from ballistic to forward flight became the focus of the testing and development program for the parafoil through a series of subscale and fullscale drops.

Testing Approach

The approach used to identify and establish the relationship of the various parameters involved in first stage inflation was to drop test a subscale version of the fullscale parafoil at the U.S. Army Yuma Proving Ground (YPG). The primary advantage of the subscale testing was that as many as eight drops could be conducted in a single day using a helicopter to deliver the test article to the release point. This allowed for rapid and relatively cheap tests, as opposed to conducting a fullscale test which typically takes a minimum of six weeks to prepare and costs at least twenty times as much to conduct as a week of subscale testing. The subscale technique also proved useful in assessing new concepts.

The subscale parafoil was faithful to the fullscale parafoil in the following parameters: airfoil shape, inlet angle, number of cells, aspect ratio and line length ratio. The subscale parafoil was 750 ft^2 in full open planform area while the fullscale X-38 parafoils are 5500 ft^2 and 7500 ft^2 in full open area. All the wings tested had a full open aspect ratio of 2.7. The subscale payload weight was calculated using mass ratio scaling to try and match the dynamics associated with deployment. The range in altitudes of subscale and fullscale testing resulted in a subscale payload weight of 930 to 980 lbs for fullscale payloads ranging from 14,000 to 25,000 lbs. The mass ratio for initial inflation during subscale testing ranged from 2.9 to 3.2, as compared to the fullscale mass ratio range of 2.9 to 3.5.

All tests were video taped using close-up and wide angle lenses on two different cameras located with orthogonal views of the anticipated release point. Very few of the tests had KTM trajectory data collected as the primary focus was the first stage deployment dynamics and loads. The riser that connects the parafoil to the load was instrumented with a riser TMS unit to measure the time history of the total load of the parachute.[2] Part way through the subscale tests, the leading edge of the lower surface of the parafoil was instrumented with leading edge TMS units to assess the spanwise spreading loads at the leading edge associated with deployment.

General Observations on Parafoil Inflation

Several factors affect how the parafoil inflates and establishes forward flight immediately out of the deployment bag: how rapidly the parafoil achieves full first stage planform spread, how rapidly the parafoil first stage is pressurized, the aspect ratio of first stage, the line length ratio of the first stage parafoil, the parafoil rigging angle and the deployment brakes. The area of the first stage parafoil, and hence the aspect ratio of first stage, was constrained by the opening loads exerted on the crew and the materials used to build the parafoil first stage. In terms of ratio of first stage to full open area, early in the program first stage was nine of the full open total of thirty one cells.

The parafoil rigging angle has a direct affect on the angle of attack at which the parafoil will trim during flight.[3] Flying at an excessively high angle of attack can result in the flow at the leading edge upper surface separating, stalling the wing. Flying at an excessively low angle of attack can result in the leading edge upper surface collapsing, depriving the wing of it's structural integrity and eventually causing the wing to stall. The larger the rigging angle, the larger the angle the canopy must transition through to go from ballistic flight to forward flight.

The magnitude of deflection of the trailing edge for deployment brakes used in first stage will determine if the parafoil will transition to forward flight at all and whether this transition is smooth or very abrupt. The amount of deployment brakes necessary are a function of the parafoil rigging angle, with the larger rigging angles requiring larger deployment brake deflections to keep the transition to forward flight from being too abrupt.

Initial Subscale Testing

The GPADS program and early X-38 fullscale tests were conducted with a line length ratio of approximately 1.0 (line length to full open span). Many of these deployments had flow separation problems during inflation and staging. This was in part due to the lack of coupling and damping between the payload and parafoil. When the canopy

surged to forward flight the resulting two body motion was of large amplitude and not damping out very rapidly. This large amplitude motion was believed to be causing the parafoil to experience a large excursion in angle of attack, on occasion causing the wing to stall. These stalls and the general softness of the parafoil disreefs were partly due to the excessively porous upper surface fabric used on these canopies, and partly due to the trim angle of attack for the parafoil at 10 degree rigging angle being too close to the stall point for this particular airfoil and inlet design.

Based on the results of the initial subscale testing early in the X-38 program the line length ratio was changed to 0.6, primarily to keep the payload more closely coupled to the parafoil motion and to stay closer to the sport industry experience. The longer line length ratio, while impervious to small parafoil dynamics during inflation exciting two body coupled motion, was very slow to damp out any two body dynamics should they occur. The longer line length also contributed to an increase in drag which hurt the overall performance of the system. The rigging angle was changed to 13 degrees to try and keep the parafoil from achieving a trim alpha that was too close to the stall angle of attack. Finally the upper surface fabric from the leading edge back 40% of the chord was treated to reduce the porosity.

Original First Stage Inflation Approach

The GPADS program and early X-38 fullscale tests attempted to control the initial shape of first stage by tying the nose together with a cut loop, with a similar cut loop used to tie the tail together. The intent was for the canopy to assume a 'football' shape during initial pressurization and then to cut the leading and trailing edge ties allowing the canopy to transition to first stage planform and eventually forward flight.

Phase 2 drop 6 (phase 2 refers to the X-38 fullscale parafoil drop tests conducted at YPG) was the first fullscale test conducted using a 13 degree rigging angle with the reduced line length ratio. For this test a GPADS 3600 ft^2 parafoil was used as an intermediate step in wing size up to the larger fullscale canopies. The nose and tail ties were kept as part of the configuration. The change in rigging angle, the treated upper surface for porosity and the decrease in line length ratio resulted in a significant improvement to parafoil deployment dynamics. The nose and tail tie approach however, proved unsuccessful in reducing the inflation dynamics. The canopy rebound at line stretch was still causing the initial first stage lower surface pressurization to be very random in terms of where it would start and how long the delay from line stretch. During this and subsequent fullscale tests, the leading and trailing edge ties were breaking before the canopy achieved the desired geometry. The canopy was passing straight through to first stage planform with the resulting surge and or charge to forward flight causing the leading edge to collapse. However, with the new rigging angle and line length ratio the canopy would recover to fully inflated condition fairly quickly and so the deployments were marginally acceptable. There was no canopy stall observed during parafoil staging but the disreefs were still very soft, with the canopy taking several seconds to become spanwise rigid following each disreef. This configuration was tested on the GPADS 7350 ft^2 canopy with similar results.

Phase 2 drop 13, the first flight of a fullscale X-38 5500 ft^2 second generation parafoil with nearly zero porosity fabric throughout, changed the deployment characteristics significantly. The new canopy was a large improvement over the original GPADS canopies in terms of overall full flight performance. The new canopy demonstrated no susceptibility to leading edge flow separation type stalls during staging and achieved a spanwise rigid state much quicker during disreefs. Unfortunately it also had the undesired characteristic of producing an initial inflation and transition to forward flight that was far more dynamic than past experience. The canopy was charging vigorously to forward flight causing two body coning motion that did not damp out until well into the parafoil staging and often time resulting in canopy twist.

In general, the fullscale tests were still experiencing a large rebound following the canopy reaching line stretch. In the worst cases this rebound was so pronounced that the inflation would take place with suspension lines above the canopy, usually resulting in burn damage to the canopy fabric (see figure 1). The large rebound was also allowing a build up in dynamic pressure contributing to the magnitude of the opening loads. As discussed in the introduction, the rebound and the randomness with which first stage was inflating appeared to be preventing any other changes from having a repeatable effect. In particular the magnitude of first stage surge to forward flight and the heading this surge would establish in was too random and too large to be acceptable.

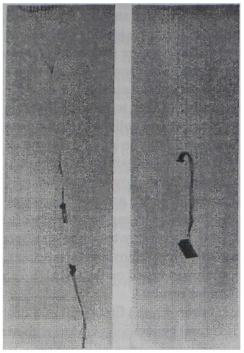


Figure 1. Fullscale 5500 ft^2 deployment which experienced a large rebound at line stretch.

Zero Reefing

A new technique was needed to gain control of the initial inflation of the canopy, allowing a low dynamic transition to forward flight. Subscale tests with permanent ties on the leading and trailing edges showed the 'football' configuration to be extremely unstable with unpredictable openings, consistent with the fullscale experience. On the subscale tests where the canopy was held in first stage to touchdown, it appeared to be inherently unstable. This is attributed to several first stage geometric factors; a low aspect ratio of 0.78, a large line length ratio of 2.14, a low arc anhedral of 2.2 degrees and deployment brakes that while adequate for the full open canopy where too low for this first stage geometry. On subscale tests where the canopy was disreefed to second stage the system would almost immediately become stable. Second stage had a total of 15 cells, an aspect ratio of 1.3 and a line length ratio of 1.3. This tendency to begin damping in second stage was consistent with the fullscale test experience.

The new technique consisted of tying a loop around the suspension lines and maintaining this loop until the first stage was rigidly inflated. The loop, also called a Rueter wrap, would subsequently be cut allowing the canopy to assume first stage. This technique was referred to as zero stage reefing. The intent was that the restrained parafoil would behave more like a round parachute during initial inflation. The concept of using a Rueter wrap was not entirely unique and in a point wise fashion reproduced the effect of a slider on a sport canopy. Four locations for the Rueter wrap were tested ranging from 14 to 48% of the chord length, as measured down from the ¼ chord point.

The subscale testing showed that the cut loop location furthest from the canopy had a minimal effect with the canopy often charging to forward flight and coning before the loop was cut. As the location of the wrap was moved up closer to the canopy, the tendency for the canopy to attempt to establish forward flight was diminished. The position closest to the floor of the canopy caused the first stage planform to be so distorted that the cells had a difficult time inflating. The two intermediate positions proved to be a good balance between performance in terms of decelerating the load and allowing first stage to transition to forward flight after releasing the Rueter wrap. During subscale testing, in all cases eventually the zero reefed system would become unstable if held long enough. Subscale testing showed a minimum of 4 seconds for subscale zero stage to start becoming unstable. This scaled up to 6 ½ seconds using mass ratio scaling for the fullscale case. The fullscale disreef to second stage was planned for 6 seconds, suggesting that any motion (associated with zero stage) would not have time to build up before disreefing to the more stable second stage. Some amount of charging was still evident in zero stage and the magnitude of turn during inflation was difficult to relate to location of the Rueter wrap. The inherent instability of first stage was not addressed because the fullscale canopy was only planned to spend two seconds between zero stage disreef and first stage disreef. The assumption was that the more stable second stage would damp out any motion that might build up during the initial transition to forward flight. The zero reefing technique was implemented on fullscale tests with the Rueter wrap located approximately 25% of chord length down the suspension lines.

The first fullscale test using the zero reefing resulted in the Rueter wrap failing as soon as it was loaded, with no improvement to the inflation dynamics. On the second fullscale test the wrap held until released as intended by the cutter. When the Rueter wrap was released the parafoil lower surface leading edge failed at the center cell allowing the canopy to split in half.

More subscale testing revealed a phenomena that had not been identified during previous subscale tests; the lower surface leading edge experiences a very large load spike immediately following the release of the Rueter wrap. In figure 2 the total parachute load (labeled 'Riser') and the spanwise

lower surface leading edge are plotted vs time for a typical zero reefed subscale test. Zero stage disreef, at approximately 11 seconds the point where the Rueter wrap is released, results in a very large spike in the leading edge spanwise loads. This load spike is associated with a momentary increase in the radius of curvature of the lower surface as the canopy assumes the new shape, that of first stage. This load spike in the leading edge following zero stage disreef is more pronounced the closer the Rueter wrap is located to the lower surface of the parafoil.

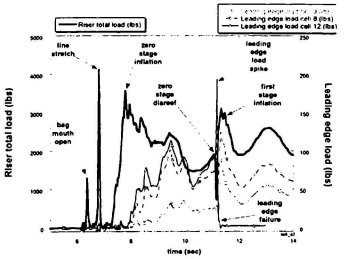


Figure 2. Typical subscale load trace using zero reefing deployment technique.

The spanwise distribution of leading edge loads for the same subscale test is plotted in figure 3. The peak load for inflation to zero stage and zero stage disreef to first stage along with the average first stage steady state load are plotted. The leading edge load spike was not observed on the subscale tests where the canopy was in backwards flight when the Rueter wrap was released.

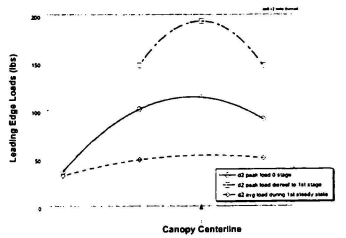


Figure 3. Subscale leading edge loads for various stages using zero reefing.

With a reinforced leading edge and the zero stage reefing rings moved further down from the canopy, the third and fourth fullscale test using this technique proved successful in terms of the Rueter wrap remaining in place until cut and not failing the parafoil lower surface leading edge. However the second of these fullscale tests (coincidentally the first flight of a parafoil deployed from a lifting body, phase 3 drop 1) transitioned to backwards flight upon releasing the Rueter wrap, recovering to forward flight during second stage with a violent collapse of the parafoil. In addition, on both tests the canopies sustained substantial burn damage to the parafoil upper surface. This was reasoned to be due to the fact that the canopy wing tips were being held lower than the center portion of the canopy, increasing the likelihood that the tip cells could begin inflating while against the interior suspension lines causing burn damage which resulted in fabric failure when the cells are pressurized.

Upper Surface Energy Modulator

While a technique that would control the transition of first stage to forward flight that did not lead to damaging the canopy was still the goal, the focus shifted to being able to achieve a repeatable first stage inflation. Again a new technique was investigated using subscale parafoil tests. The basic concept of the new technique was to maintain tension on the central upper surface of the canopy during initial presentation to the freestream. The implementation of this technique involved attaching the center cell of the canopy, at about the third chord point, to the deployment bag using an energy modulator that when completely stroked would release the canopy.[4]

In addition to the parafoil upper surface energy modulator technique, several changes to first stage geometry were implemented. Two cells were added to first stage (taken from later stages) along the center line and the trailing edge of the central three cells of first stage where allowed to "fly free" without any deflection for deployment brakes (referred to as a vented trailing edge). These additional cells increased the aspect ratio of first stage to 0.95, decreased the effective line length ratio to 1.74 and increased the arc anhedral to approximately 8 degrees. The vented trailing edge was added to encourage the canopy to establish and maintain forward flight; the thought being that a strong surge to forward flight was the lesser of two evils when compared to backwards flight in first stage. The vented trailing edge was also a change in the direction of how the deployment brakes for sport

industry parafoils are implemented. Unlike the sport parafoil, the X-38 parafoil had deployment brakes across the entire trailing edge.

These changes proved extremely successful in attaining a repeatable presentation of the canopy to the freestream during subscale testing. In particular, this technique appeared to take advantage of the canopy tendency to rebound at line stretch. By holding the central section with the modulator the tips would rebound down and away promoting the spread of the canopy into the first stage planform. This quicker spreading of first stage allowed the drag area to build up more rapidly, maintaining tension on the suspension lines almost throughout the first stage deployment. Figure 4 is a comparison of the rebound for a subscale deployment with and without implementing the upper surface energy modulator.

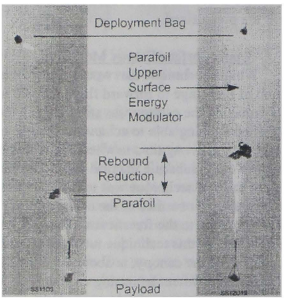


Figure 4. Comparison of subscale rebound at line stretch with and without the parafoil upper surface energy modulator assisted deployment.

Several series of subscale tests were performed using the parafoil upper surface energy modulator, investigating the effect of rigging angle (ranging from 4 to 16 degrees) and deployment brake setting (ranging from 14 to 32% chord) on first stage opening loads and transition to forward flight.

In general, the following observations were made related to the first stage deployment brake settings and inflation dynamics. If the brakes are too deep the canopy will actually fly backwards with the curvature of the trailing edge brakes effectively serving as a leading edge. There appears to be no true trim point for this condition. The payload and parafoil will develop a coupled motion oscillating back and forth in the pitch plane, often time developing into a coning motion. For marginally too deep brakes it has been observed that the first stage parafoil will trim into a very steep but stable flight path angle. As the canopy disreefs and additional cells are presented, the increase in wing aspect ratio and decrease in system line length ratio allow the parafoil to transition to normal forward flight. This has been observed in both subscale and fullscale tests. The transition to forward flight may cause the leading edge to fold down and the chord to deform momentarily, but no worse than an opening that immediately transitions to forward flight with partially pressurized cells. However for a canopy that initially establishes backwards flight, the transition to forward flight will almost certainly result in a temporary collapse of the parafoil. Not unlike an inflation that transitions to forward flight with a very hard surge or charge, the leading edge will be overrun by the canopy. This will result in a large deformation of the canopy planform area in the chordwise direction. After the peak point in the surge the flow along the upper surface is reestablished, the spanwise forces begin to assist in spreading the canopy and the parafoil internal pressurization returns the wing to it's intended geometry. This transition from backwards to forwards flight will usually excite a large coupled parafoil to payload motion that easily transitions to coning.

During the initial drops using the upper surface energy modulator, it was decided that the strong trailing edge tie was no longer needed to control the trailing edge as the parafoil is deployed. The trailing edge tie appeared to hinder the effectiveness of the deployment brakes during initial inflation. As a result, the majority of the subscale tests with the upper surface energy modulator were conducted using a weak break tie on the trailing edge, and this was only to facilitate packing the canopy. By holding the canopy in first stage all the way to impact; it was observed that the new first stage geometry, with brakes optimized to minimize surge, would establish stable forward flight.

The low rigging angle tests, primarily at 4 degrees, required very little deflection of the trailing edge for deployment brakes and had a very subtle transition from ballistic to forward flight. However, for the modified Clark-Y airfoil used on the X-38 parafoils, the 4 degree rigging angle resulted in a high trim angle of attack that was very close to the stall point which made the wing susceptible to collapsing when encountering thermals rising from the desert floor. In addition the lower rigging angles tended to be less stable in roll developing a roll oscillation, which while not divergent would not

damp out either. The mid range rigging angles, 7 and 10 degrees, resulted in larger surges during transition to forward flight but also trimmed at lower angles of attack than at 4 degrees rigging and did not exhibit the tendency to experience roll oscillations. These two rigging angles appeared to have the best trade off between a reasonable surge to forward flight and maintaining some margin between the deployment brakes that prevented large surges and the brakes which resulted in backwards flight. The larger rigging angles, 13 and 16 degrees, require the canopy to traverse through a larger angle change from ballistic to forward flight and generally resulted in very large surges to forward flight. Indeed, for 16 degrees rigging angle it was difficult to control the surge without the resulting first stage flight path angle being either excessively steep or backwards.

During previous subscale test series, it had been difficult to relate the trend of opening loads to the configuration due to the randomness in the parafoil first stage inflation following the rebound at line stretch. With the very repeatable canopy presentation associated with the upper surface energy modulator assisted deployment, it became obvious that the drops that charged to forward flight tended to have lower opening loads. From the riser load trace it appears that these openings spread the opening load out over a larger period of time. Those tests where the canopy surged very little always tended to have quicker inflation times and higher opening loads. In most cases the surge to forward flight was taking place with the canopy not fully pressurized, with the outboard cells usually the softest. This would result in the outboard trailing edge surfaces momentarily colliding at the peak of the surge. The soft end cells was most pronounced for the larger rigging angle tests. The canopy always regained the first stage planform following the surge. In general, the new technique was resulting in a more rapid first stage inflation.

As a result of these series of subscale tests the rigging angle of 10 degrees along with the parafoil upper surface energy modulator and first stage geometry changes were implemented on the fullscale parafoil tests. The brake setting used for subsequent tests of the fullscale canopies was scaled up based on the %chord brake deflection deemed the best setting from the subscale tests. The length of the modulator for the fullscale application was scaled such that it would release the canopy at the same point in the inflation as in the subscale tests. This involved estimating the line stretch velocity and inflation time for fullscale first stage. The energy modulator load for the fullscale application was scaled by taking the ratio of the weight of the fullscale canopy to the subscale canopy weight multiplied by the subscale energy modulator load and then rounding to the nearest multiple of available energy modulators.

The first fullscale test implementing the parafoil upper surface energy modulator was flown on phase 2 drop 22 using an X-38 5500 ft^2 parafoil (see figure 5). The benefits observed in the subscale tests were in large part realized in the fullscale tests. One exception was the rebound at line stretch on the fullscale drops which while a significant improvement over past fullscale drops, appeared more pronounced than the subscale experience. This is most likely due to the fact that the mass of the subscale canopy could not be scaled. This is apparent in the ratio of canopy mass to canopy first stage area; for the subscale tests it was 0.19 lb/ft^2 while this ratio for the fullscale canopy was more like 0.31 lb/ft^2. A total of four fullscale pallet tests and three lifting body parafoil deployments where perfomed using this technique.

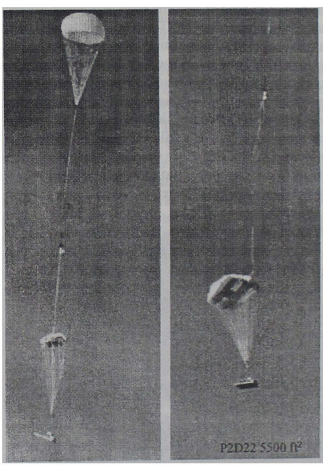


Figure 5. First fullscale parafoil upper surface energy modulator assisted deployment P2D22.

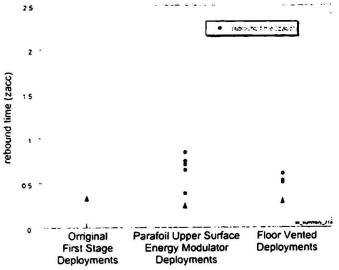


Figure 6. Rebound times for fullscale parafoil deployment for the various techniques tested.

Floor Inlets

While the canopy was achieving a good repeatable initial spanwise spread with the upper surface energy modulator technique, it was not getting enough time to become rigid before the transition to forward flight. This lack of internal pressurization was still resulting large canopy deformations during the surge which were thought to be contributing to the off heading openings and twist being experienced in first stage (see figures 7 and 8). While experience indicated that the canopy would always recover, there was still a strong desire to pressurize the canopy before transition to forward flight, primarily in hopes that this improvement might eliminate twist all together.

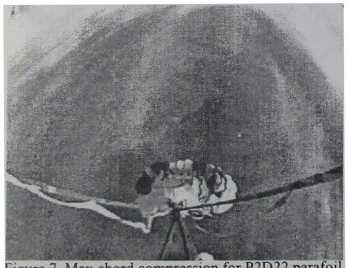


Figure 7. Max chord compression for P2D22 parafoil upper surface energy modulator assisted deployment of 5500 ft^2 parafoil.

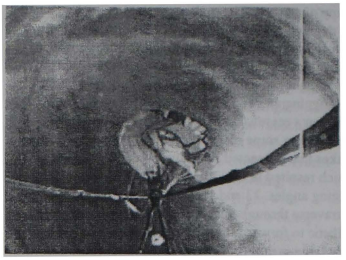


Figure 8. Max chord compression for P2D23 parafoil upper surface energy modulator assisted deployment of 5500 ft^2 parafoil.

An additional modification to first stage was tested in subscale parafoil tests to try and achieve a rigid first stage prior to surge. This technique involved opening holes in the floor panels of first stage cells to assist in pressurization of the canopy. The floor vents would have mesh in place of the zero porosity fabric, with flaps of fabric on the inside that would prevent the canopy once pressurized from deflating. The area of these first stage floor inlets was built to match the projected leading edge inlet area at 90 degrees flight path angle. The floor inlets were located such that the pressurization through the nose inlets and floor inlets would achieve a rigid canopy as quickly as possible.

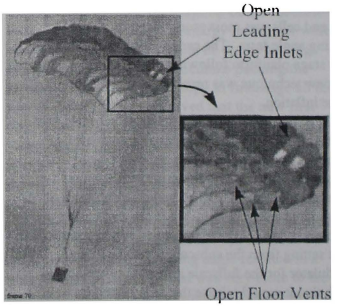


Figure 9. Subscale inflation with floor vents.

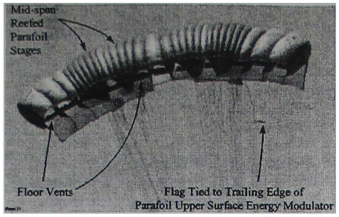


Figure 10. Head-on view of subscale parafoil first stage with floor vents closed off.

This technique proved to be another huge leap (along with the upper surface energy modulator) towards controlling first stage inflation and transition to forward flight. With the canopy no longer limited to pressurizing through the leading edge inlet, by the time the canopy surged to forward flight it was almost completely pressurized. Based on the subscale tests, the first stage opening loads had increased by as much as 90% and the peak leading edge spanwise load associated with first stage inflation increased by almost 80% over that measured using the upper surface energy modulator alone deployments.

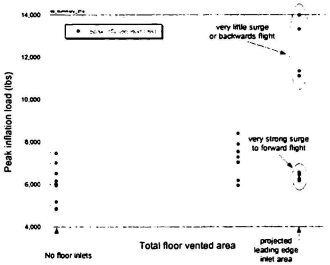


Figure 9. Subscale peak first stage load with and without floor vents.

The floor vented first stage was first tested using an X-38 5500 ft^2 parafoil on phase 2 drop 27 (see figure 10). Because of structural limitations of the canopy, only half the floor vent area that was considered optimal based on the subscale testing could be opened. Regardless, the result was a significant improvement in the pressurization of the canopy with the compression of the chord related to the surge to forward flight reduced, just as had been observed during the subscale tests. The magnitude of the canopy rebound at line stretch was also reduced (see figure 6).

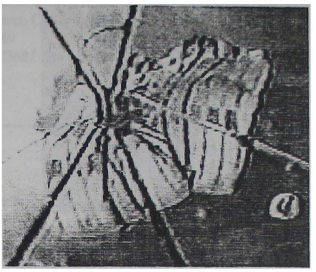


Figure 10. Max chord compression for P2D27 with ½ floor inlets open and upper surface energy modulator assisted deployment 5500 ft^2 parafoil.

The new X-38 7500 ft^2 parafoil, the design intended for use on the first spacecraft test, has been flown twice to date with all floor vents opened at the spacecraft wing loading. The results of these two drops was very consistent with the one test of the X-38 5500 ft^2 canopy and the subscale tests with floor vents. No indication of canopy compression was noted (figure 11) in the first stage at the maximum point in the surge during the two 7500 ft^2 tests. In all cases, subscale and fullscale magnitude of twist during deployment was significantly reduced using the floor vent assisted inflation technique.

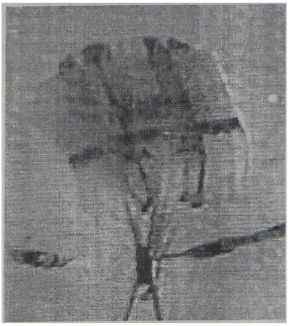


Figure 11. Max chord compression for P2D28 with complete floor inlets open and upper surface energy modulator assisted deployment of 7500 ft^2 parafoil.

Conclusions

The X-38 program has spent several years working the issues associated with deploying large parafoils capable of recovering loads up to 24,000 lbs. The desire to achieve a low dynamic on heading opening has led to several innovations that have greatly improved the repeatability with which first stage assumes full spread and becomes rigid. The most significant of these has been the parafoil upper surface energy modulator and the first stage floor vents. The opening loads and dynamics of the parafoil deployment and transition from ballistic to forward flight have been minimized to meet the constraints of not harming an ill and deconditioned crew member. The magnitude of twist during the parafoil deployment phase has been reduced to less than 180 degrees for the three fullscale flights implementing the floor vents. The improvements made to first stage deployment will be tested over the next few years in order to human rate the parafoil system. These tests will create an experience base with which to quantify just how well these improvements can stand up to the randomness inherent to deploying a 1,400 lb parafoil.

Acknowledgments

The author would like to extend thanks to the folks that spent countless weeks and months over the course of several years participating in the subscale and fullscale tests. In particular, thank you for supporting the subscale tests which invariably only took place in the heat of the Arizona summer.

References

[1] Machin, R., Stein, J. and Muratore, J., "An Overview of the X-38 Prototype Crew Return Vehicle Development and Test Program", AIAA-99-1703, June 1999.

[2] Machin, R., Fitzgerald, S. and Royall, P., "Technique for Measuring Parafoil Deployment and Steady State Loads in the Dispersion Risers and Leading Edge Reinforcement Tape", AIAA-99-1733, June 1999.

[3] Iacomini, C. and Madsen, C., "Investigation of large Scale Parafoil Rigging Angles: Analytical and Drop Test Results", AIAA-99-1752, June 1999.

[4] Bennett, T., Smith, J. and Fox, R., "Testing and Development of the NASA X-38 Parafoil Upper Surface Energy Modulator", AIAA-99-1753, June 1999.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37263

AIAA-2000-4388

AN OBJECT-ORIENTED DESIGN FOR TRIM

Jason R. Neuhaus*

Unisys Corporation
NASA Langley Research Center
Mail Stop 169
Hampton, VA 23681

Abstract

This paper presents an Object-Oriented Design for trimming dynamic models. By applying Object-Oriented Programming techniques to trim, in conjunction with an object-oriented model structure, generic trim rules can be developed and shared between different model types. This trim design can be used to trim many types of models, including automobiles, aircraft, spacecraft, and rotorcraft. Class hierarchies and interactions between the model, trim, and trim rule classes, as well as a general overview of Object-Oriented Programming, are presented.

Introduction

The objective of a trim routine in simulation is to modify the states of a vehicle until equilibrium is reached. For aircraft, this equilibrium is typically defined as the set of states for which all body axis translational and angular accelerations are zero. Trim is used to properly initialize an aircraft so that flight can begin in any flight situation and maintain an initial set of specified states after the simulation switches into run/operate mode. For example, trimming on the ground involves adjusting the height of the aircraft until the vertical acceleration reaches zero. This is done so the aircraft does not fall, or the landing gear are not so compressed to cause the aircraft to jump, when the simulation is switched into the operate mode. For example, the aircraft throttle(s), aileron, rudder, and elevator can be adjusted so that the aircraft maintains straight and level flight with no pilot inputs after the simulation begins running.

*Aerospace/Software Engineer, Member

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

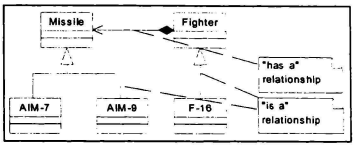


Figure 1 Simple UML Class Diagram

Object-Oriented Design

An Object-Oriented Design is made up of classes. Each class encapsulates a well-defined set of tasks and provides an interface for other classes to interact with it. Each class contains a set of methods (functions) and member data that define the behavior of the class. Through encapsulation, the class becomes a black box where only a public interface is available. This allows classes to limit the access of methods and member data to other classes. Methods and data can be defined as public, protected, or private. Public methods are accessible to any class. Protected methods only allow derived classes to access the method. Private methods and data are accessible only to that class.

In addition to defining their own methods, classes can share and overload† the methods of other classes through use of inheritance. Inheritance defines a relationship between classes, wherein one class shares the structure and/or behavior of another class [1]. The parent class is the class from which inheritance is taking place. The class that is inheriting from the parent class is the derived class. Inheritance is often referred to as an "is a" relationship. Figure 1 shows a simple Unified Modeling Language (UML)[2] class diagram depicting the relationships between a fighter and missile classes. The AIM-9 is a derived class, which inherits from the Missile class, the parent class. An AIM-9 "is a" missile.

† redefine or add additional functionality

An object is a particular instance of a class. When an object is created, it maintains all of its own data separate from all other objects of the same type[‡]. In this way, an arbitrary number of copies of a class can be created and used without data being mixed between the different instances of a particular class. Multiple fighter objects can be created, where each maintains its own private data members, such as altitude, orientation, accelerations, etc.

Any place where a parent class is used, a derived class can be substituted because of inheritance. For example, if a Fighter class contains Missiles, "has a" relationship, any class that derives from the Missile class can be used in place of the Missile class used by the Fighter. An AIM-9 can be used because the AIM-9 "is a" missile. The AIM-9 class contains all the functionality of the Missile class, plus the additional functionality specific to that particular type of missile. The Fighter class can only call public methods that are declared in the Missile class. Additional methods declared in the derived classes of the Missile class cannot be accessed, because the parent class has no knowledge of the data and methods declared in classes that derive from it.

Polymorphism is the overloading of a method of a parent class in a derived class. It is used to extend or replace functionality in the parent class. Methods that can be overloaded are marked by the virtual keyword in C++. The correct overloaded method to call is decided at run time. The correct method to call is not decided at compile time because the method to call depends on the exact type of the object being accessed, which, because of inheritance, may be the parent class, or any classes that derive from it. For example, assume the Missile class contains a public virtual method named *update* that performs some function. The AIM-7 class overloads the *update* method to add AIM-7 specific code, but the AIM-9 class does not overload *update*. If the fighter object has both an AIM-9 and an AIM-7 object, and calls *update* on both, the *update* method from the Missile class will be called for the AIM-9 object, but the overloaded *update* method will be called for the AIM-9 object. This occurs even though the fighter object has no knowledge what types of missile objects are being used.

[‡] Data *can* be shared between instances of a class, but is beyond the scope of this paper.

LaSRS++ Trim Classes

The following classes are used in the Langley Standard Realtime Simulation in C++ (LaSRS++)[3] trim design and are shown in Figure 2.

Vehicle Class

A vehicle's states propagate through time in reaction to external forces and moments. The forces and moments determine the translational and angular accelerations of the vehicle.

Aircraft Class

The Aircraft class inherits from the Vehicle class. An aircraft is a vehicle that flies through an atmosphere. The dynamics of the aircraft are primarily influenced by aerodynamic, engine, and gear forces and moments.

TrimRule Class

The TrimRule class defines relationships between trim states, errors, and gains used to trim a vehicle object. The trim rule states and errors are based on vehicle data. Trim states are modified based on errors and gains, until the error is within a specified tolerance. Each trim rule can have multiple sets of states, errors, and gains as well as constraints. A constraint modifies a state to a predetermined value that can be a function of other states or a constant value.

AircraftTrimRule Class

The AircraftTrimRule class inherits from the TrimRule class. It adds the ability to access aircraft data, in addition to vehicle data, for use in trim states and errors. Some of the LaSRS++ aircraft trim rules are listed at the end of this paper.

Trim Class

The Trim class is responsible for performing all the basic trim operations on a vehicle. These operations include selecting trim rules, checking for rule conflicts, setting target values for body translational and angular accelerations, and finding a trim solution.

A trim solution is found by operating on a list of trim rules until a solution is found or problems finding a solution are detected. Some of the conditions checked to determine if a solution could not be found are if the iteration count reaches a preset maximum and no solution is found, or an error value reaches a steady state value that is not within the specified tolerance. In the event of a problem locating a solution, trim states and errors are output to aid in determining which states had trouble trimming.

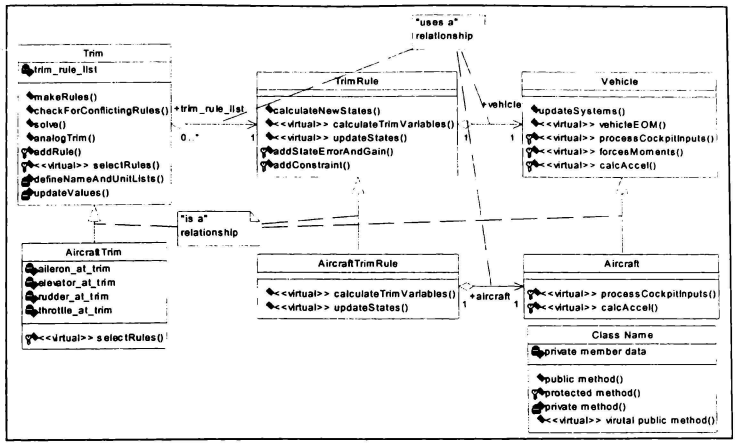


Figure 2 LaSRS++ Trim UML Class Diagram

AircraftTrim Class

The AircraftTrim class inherits from the Trim class. It selects aircraft trim rules based on the desired aircraft initialization. Four additional states exist in the AircraftTrim class for the purpose of overriding cockpit values while trimming. These four data members are *aileron_at_trim*, *elevator_at_trim*, *rudder_at_trim*, and *throttle_at_trim*. These are used as generic inputs for roll rate, pitch rate, yaw rate, and propulsive thrust respectively. Each different aircraft maps the four generic inputs to inputs specific to that aircraft. Depending on the control system, *elevator_at_trim* may drive the elevator position directly, and a separate trim variable drives the pitch stick. This is often done for α, Nz, and pitch rate command systems. Once the aircraft is trimmed, any offsets in control positions can be back driven into the hardware, displayed so the pilot can match the inputs, or cockpit inputs can be biased based on the trimmed values.

The AircraftTrim class does not contain any logic for trimming. All of the trimming logic is contained in the parent class, Trim. Through inheritance, the AircraftTrim class can use these same methods. In this way, the methods in Trim can be used on both vehicles and aircraft.

Trim Algorithm

Each simulation object uses either a trim or an aircraft trim object to trim itself, depending on the object type. When the simulation is switched into trim mode, all the simulation objects are trimmed in sequence. The following sections give the basic outline of the trimming procedure.

Rule Setup

Before the simulation objects are trimmed, each must setup initial values and chose the rules that will be used to trim the object. Typically, rules are chosen that zero out all body translational and angular accelerations. How each of the body translational and angular accelerations are trimmed to zero is dependent on the rules that are selected. It is sometimes desirable to trim an acceleration term to a known non-zero value, or not to trim it at all. Trimming to non-zero accelerations is often done to match old data or third-party test cases. An acceleration term might not be trimmed at all on an aircraft without any engines; where there is no thrust to trim out the body-x axis acceleration.

Trim Loop

After the trim rules have been setup, the solution process begins. A trim solution is found by looping until a solution is found, a maximum iteration count has been reached, or the simulation exits trim mode. A maximum iteration count is hit when one or more error signals did not converge to zero. This usually occurs when an error does not converge fast enough, converges, but oscillates around zero, or reaches a steady state value that is not within the specified tolerance. An error that converges too slowly usually indicates that the magnitude of a gain that was too small. An error that converges quickly, but oscillates around the solution, typically results when the magnitude of a gain was too large. Warnings are printed if trim believes it has encountered a steady state error. A steady state error usually indicates that one or more variables have reached a limit, for example, a control surface deflection limit.

After the simulation object has been successfully trimmed, the trim routine exits and the aircraft is prepared to operate at the flight condition specified.

Methods

Figure 3 shows a UML sequence diagram[2] listing the events that occur when trimming an aircraft. The methods called are described below.

Trim Methods

makeRules

Before trim is initiated, the user selects the trim rules to be used. This forms the *trim_rule_list* on which the trim object will operate.

selectRules

This method is called by the *makeRules* method and is responsible for determining which rules the user has selected. Once the rules have been identified, they are added to the *trim_rule_list* using the *addRule* method.

defineNameAndUnitLists

Once the list has been formed, the names and units of the states, errors, and gains of all rules on the list are gathered and stored. The names will be used to ensure that two conflicting rules are not selected, and for output purposes.

checkForConflictingRules

This method warns the user if two conflicting rules are selected. Conflicting rules may prevent the vehicle from trimming or may result in a condition where multiple solutions exist. Two rules conflict if either the same state is modified or the same error is monitored by multiple rules. No preventive action is taken in the event of a conflict. Combinations of states and errors that do not directly conflict are not detected. For example, selecting trim rules that specify angle of attack, pitch angle, and flight path angle values would cause a conflict but not be detected.

solve

Before trimming begins, the simulation object is set up at the user-defined conditions. This is accomplished by having each trim rule initialize its internal states to the simulation object's initial states. Then all the trim rules update the states in the simulation object with their internal states, including any constraints. This ensures that constraints are set up properly in the first pass, ands is accomplished through calls to the trim rule methods, *calculateTrimVariables* and *updateStates*.

updateValues

Calling this method copies all the trim state, error, and constraint values from the individual trim rules into the trim object. This is done so that the data can be recorded for diagnostic purposes.

analogTrim

This method is called by the *solve* method and performs one iteration in the trimming process. Each iteration involves updating the equations of motion, calculating new trim variable values, and pushing the new trim states back into the vehicle from the rules.

Aircraft Trim Methods

selectRules

The AircraftTrim *selectRules* method overloads the virtual *selectRules* method of the Trim class. This method adds logic for selecting aircraft trim rules based on the user's specifications. The aircraft trim rules selected are added to the rule list through the *addRule* method.

Trim Rule Methods

calculateTrimVariables

The *calculateTrimVariables* method updates the states, errors, and gains for each trim rule selected. The states are gathered from the current set of vehicle data. The errors are computed based on the deviation of vehicle data from target values. Gains are updated from the associated trim object.

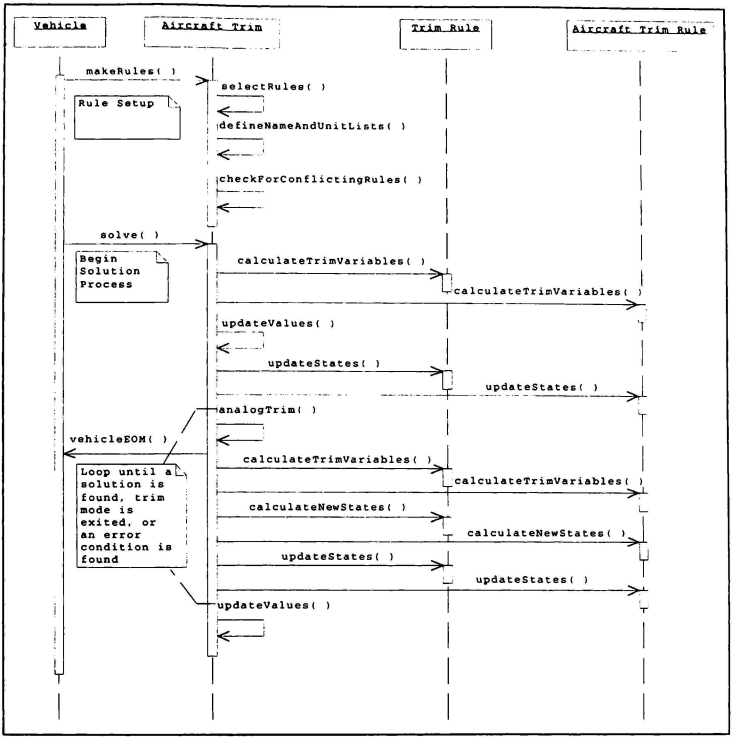


Figure 3 Trim UML Sequence Diagram

calculateNewStates

This method calculates new trim states. The following formula is normally used to calculate the new states based on the states, errors, and gains calculated in the previous iteration. Other methods of calculating the next set of trim states can be added by overloading this method.

$$state = state + error \cdot gain$$

Using this formula to calculate new trim states results in a very stable solution process, but may take longer to converge on a solution than some other trim methods.

updateStates

This method updates vehicle data that correspond to the trim states of the rule. Any constraints contained in the trim rule are recalculated and vehicle data is set accordingly.

Aircraft Trim Rule Methods

The AircraftTrimRule class does not add any additional methods to the TrimRule class. The only change is in the call to the constructor. When constructed, an aircraft trim rule object uses an aircraft object and an aircraft trim object argument, as opposed to a vehicle object and a trim object used by the TrimRule class.

calculateTrimVariables

The *calculateTrimVariables* method updates the trim states, errors, and gains for each aircraft trim rule selected. The states are gathered from current aircraft or vehicle data. The errors are computed based on the deviation of aircraft data from their corresponding target values. Gains are updated from the associated aircraft trim object.

updateStates

This method updates aircraft data based on trim states. Any constraints contained in each aircraft trim rule are recalculated and set in the aircraft.

Vehicle Methods

vehicleEOM

Several processes take place when the equations of motion of a vehicle are updated. These include processing cockpit inputs, recalculation of derived data, calculation of accelerations, and updating vehicle systems. The equations of motion are also updated each iteration while the simulation is running. During trim, a vehicle does not integrate any accelerations to determine the new set of vehicle data. The new vehicle data are determined by the trim rules.

Processing cockpit inputs updates the values of the cockpit inputs into the vehicle. In trim, however, cockpit inputs are typically overridden by internal trim variables.

Derived data are recalculated to make sure the vehicle has a consistent set of states. For example, making sure angle of attack (α), pitch (θ), and flight path angle (γ) are consistent for aircraft.

Accelerations are calculated from the vehicle data. The accelerations of interest to trim are typically body axis translational and angular accelerations.

The vehicle systems are updated so that any data in the systems are consistent with the current frame. Vehicle system updates can occur after the derived data calculations or after the acceleration calculations, depending on the nature of the individual vehicle system. Some examples of vehicle systems include fuel systems, control systems, and propulsion systems.

Aircraft Methods

The differences between the Vehicle and Aircraft classes exist in methods called by the *VehicleEOM* method. The Aircraft class overloads the methods called by *VehicleEOM*, so that aircraft trim control variables can be properly mapped to the aircraft cockpit. For the four control variables used by aircraft trim (see AircraftTrim Class), each aircraft individually determines which trim control states override which cockpit inputs or other aircraft data. Aircraft also have additional derived data, which are calculated.

Filters and Integrators

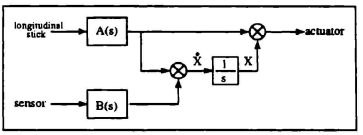


Figure 4 Pitch Loop Integrator

Filters and integrators require special attention when trimming vehicles. It is desirable, during trim, for filters to respond with their steady state outputs. Two different behaviors for integrators are used in trim. In the LaSRS++ framework, every filter and integrand knows the current mode of the simulation.

When the simulation is in trim mode, filters set their outputs to their steady state output values, $t = \infty$ ($s = 0$ in the Laplace domain). In this way, all internal derivatives of the filter are zero, such that, when the vehicle switches from trim to operate mode, there are no undesired transients.

Integrators can behave in two different manners during trim. If an integrator has a known initial value and derivative, the integrator can be held at the initial value and derivative. An example of this would be a mass integrator for burning fuel. The initial fuel amount is specified, and the initial burn rate can be directly computed from the throttle settings of the engines. If allowed to integrate during trim, the vehicle would burn fuel before it ever went into operate.

The second type of behavior is for integrators that must have derivatives equal to zero, or some other known value, but whose initial value is not known. In this

case, a trim rule must be added to monitor the integrators derivative until the target value is reached. An example of this is a pitch loop integrator is shown in Figure 4.

Earth Models

The LaSRS++ framework has three different earth models, flat, round, and ellipsoidal. The earth can also be rotated for the round and ellipsoidal models. Since non-flat earth models can be used, the assumption that trimming ailerons and elevator positions to zero will make the roll and pitch rates zero is invalid. Depending on the location of the vehicle relative to the world and the heading, small elevator, aileron, and rudder inputs may be needed to zero out pitch, roll, and yaw accelerations relative to the earth's surface, due to the curvature and rotation of the earth.

Results

This run was conducted at an altitude of 10,000 feet, target Mach number of 0.5, in a coordinated turn at a 10 degree bank angle, with a target ground track of 45.0 degrees. A rotating ellipsoidal earth model with inverse radius squared gravity was used. The aircraft trim rules used were Velocity Mach, Pitch Rate, Flight Path, Alpha Trim, Coordinated Turn, and Lateral Surfaces. The error tolerance was set to 1E-10 and took 900 iterations to find a solution. The resulting aircraft states are listed in Table 1. Plots of interesting trim states and errors vs. iteration can be found at then end of this paper.

Velocity Mach

The Velocity Mach rule adjusts Vtotal until the target Mach number (0.5) is reached. This rule makes sure the Mach number specified is correct during trim, even if another rule trims the altitude.

Alpha Trim

The Alpha Trim rule adjusts the angle of attack until the body-Z axis acceleration is zero.

Flight Path

The Flight Path rule is responsible for setting up the throttle and the pitch angle of the aircraft. The throttle is adjusted until the body-x axis acceleration is zero. The pitch angle is adjusted until the specified flight path angle (0.0) is reached.

Coordinated Turn

The Coordinated Turn rule trims the aircraft into a steady turn, v dot equal to zero. This is accomplished by adjusting the sideslip until the body sideward acceleration, v dot, reaches zero. The yaw angle is trimmed until the desired ground track angle (45.0 degrees) is reached.

Lateral Surfaces

The Lateral Surfaces rule adjusts the *aileron_at_trim* and *rudder_at_trim* values to zero out the body roll and yaw accelerations respectively.

Pitch Rate

The Pitch Rate rule adjusts the *elevator_at_trim* until the body pitch rate is zero.

Table 1 Trim Results

h	Altitude	10,000 ft
M	Mach	0.5
Vt	Vtotal	538.70 ft/sec
IAS	Indicated Airspeed	276.85 knots
qbar	Dynamic Pressure	254.73 psf
α	Angle of Attack	5.0818 degrees
β	Sideslip	-0.25069 degrees
hdot	Altitude rate	7.47e-12 ft/sec
u	Body-X velocity	536.58 ft/sec
v	Body-Y velocity	-2.3570 ft/sec
w	Body-Z velocity	47.717 ft/sec
p	Body roll rate	-0.0518 deg/sec
q	Body pitch rate	0.1021 deg/sec
r	Body yaw rate	0.5875 deg/sec
ϕ	Roll	10.0 degrees
θ	Pitch	4.9617 degrees
ψ	Yaw	46.128 degrees
pdot	Body roll acceleration	6.70e-12 deg/sec^2
qdot	Body pitch acceleration	-6.65e-10 deg/sec^2
rdot	Body yaw acceleration	6.70e-11 deg/sec^2
udot	Body X acceleration	-1.39e-12 ft/sec^2
vdot	Body Y acceleration	-1.56e-11 ft/sec^2
wdot	Body Z acceleration	-2.45e-12 ft/sec^2
δa	*aileron_at_trim*	-0.4942 non-dim
δe	*elevator_at_trim*	-3.1240 non-dim
δr	*rudder_at_trim*	1.2146 non-dim
δt	*throttle_at_trim*	0.1426 non-dim

Conclusions

There are many advantages to using an object-oriented trim design. In an object-oriented framework, a set of generic trim rules can be developed that work on any type of simulation object. For aircraft, generic aircraft trim rules can be used. Any aircraft that derives from the Aircraft class, 757, F-16, etc., can all use the same rules for trimming a given condition. Through the

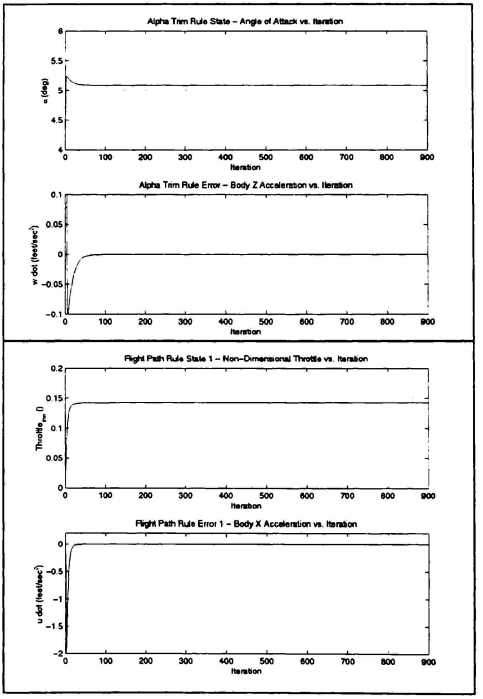
addition of a generic method (function) to check for rule conflicts, complex sections of code to check for specific rule conflicts are no longer needed and no longer need to be maintained when a new rule is added. The only vehicle specific code required is to map the generic control variables into the correct cockpit input on the vehicle.

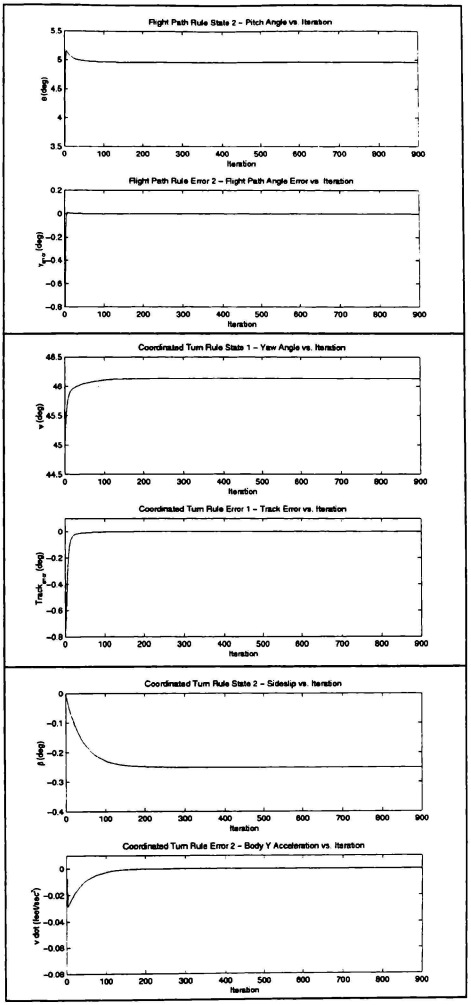
References

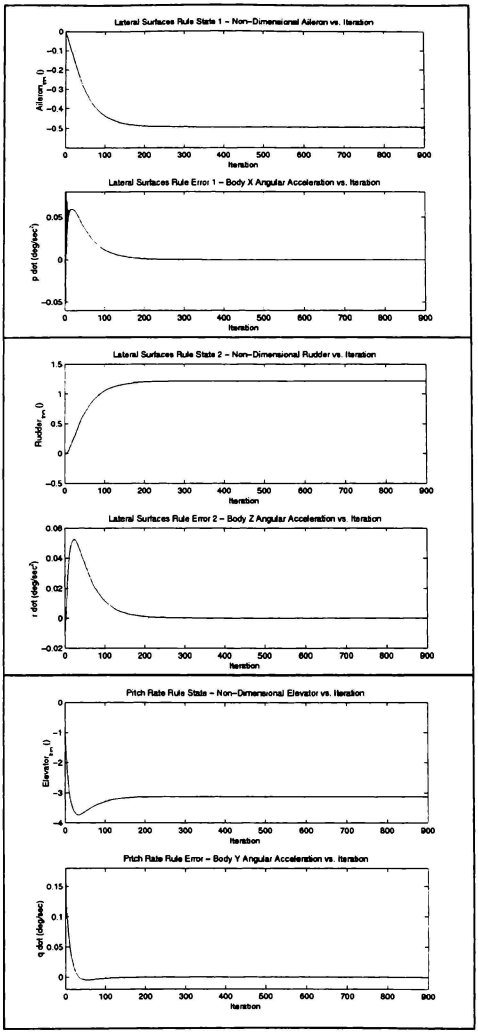
1 Booch, Grady. *Object-Oriented Analysis and Design With Applications*. The Benjamin/Cummings Publishing Company, Inc., Redwood City, California, 1994.

2 Muller, Pierre-Alain. *Instant UML*. Wrox Press Ltd. Chicago, Illinois, 1997.

3 Leslie, R.; Geyer D.; Cunningham, K.; Madden, M.; Kenney, P.; Glaab, P. *LaSRS++: An Object-Oriented Framework for Real-Time Simulation of Aircraft*. AIAA 98-4529, Modeling and Simulation Technology Conference, Boston, MA, August 1998.




Flight Path Rule State 2 – Pitch Angle vs. Iteration
θ (deg)
Iteration
Flight Path Rule Error 2 – Flight Path Angle Error vs. Iteration
γ_err (deg)
Iteration
Coordinated Turn Rule State 1 – Yaw Angle vs. Iteration
ψ (deg)
Iteration
Coordinated Turn Rule Error 1 – Track Error vs. Iteration
Track_err (deg)
Iteration
Coordinated Turn Rule State 2 – Sideslip vs. Iteration
β (deg)
Iteration
Coordinated Turn Rule Error 2 – Body Y Acceleration vs. Iteration
v dot (feet/sec²)
Iteration


Lateral Surfaces Rule State 1 – Non-Dimensional Aileron vs. Iteration
Aileron_nm ()
Iteration
Lateral Surfaces Rule Error 1 – Body X Angular Acceleration vs. Iteration
p dot (deg/sec²)
Iteration
Lateral Surfaces Rule State 2 – Non-Dimensional Rudder vs. Iteration
Rudder_nm ()
Iteration
Lateral Surfaces Rule Error 2 – Body Z Angular Acceleration vs. Iteration
r dot (deg/sec²)
Iteration
Pitch Rate Rule State – Non-Dimensional Elevator vs. Iteration
Elevator_nm ()
Iteration
Pitch Rate Rule Error – Body Y Angular Acceleration vs. Iteration
q dot (deg/sec)
Iteration

Aircraft Trim Rules

All of the following rules derive from the AircraftTrimRule class. See Table 1 for symbol definitions.
Notation: θ,δt = target – indicates Pitch and *throttle_at_trim* are set to target values.

Velocity Rules	**State**	**Error**	**Constraints**
Velocity Drag	Vtotal	u dot	
Velocity Mach	Vtotal	mach	
Velocity Speed	Vtotal	speed	
Velocity Alpha	Vtotal	w dot	
Alpha Rules	**State**	**Error**	**Constraints**
Alpha Trim	α	w dot	
Constant Alpha			α = target
Loaded Loop	α	Nz	q = f(w dot world relative , u air relative)
Longitudinal Rules	**State**	**Error**	**Constraints**
Constant Pitch	δt	u dot	θ = target
Constant Pitch/ Constant Thrust			θ,δt = target
Constant Thrust	θ	Vtotal dot	δt = target
Constant Thrust, Pitch	θ	w dot	δt = target
Constant Thrust, Rate of Climb	θ	h dot	δt = target
Flight Path	δt	u dot	
	θ	γ	
Pitch Thrust	θ	w dot	
	δt	u dot	
Rate of Climb	δt	u dot	
	γ	h dot	
Lateral Rules	**State**	**Error**	**Constraints**
Bank into Wind	ψ	track	β = f(u, ϕ, Vtotal)
	ϕ	v dot	
Coordinated Turn	ψ	track	ϕ = target
	β	v dot	ϕ,θ,ψ dot = f(g, θ, ϕ, u, w)
Crab into Wind	ψ	track	
	β	v dot	
Loaded Roll	ψ	track	ϕ,θ,ψ dot = f(g, θ, ϕ, u, w)
	β	v dot	
	ϕ	u dot	
No Lateral Motion			ψ = target
			δa, δr, ϕ, ϕ dot , θ dot , ψ dot = 0
One Engine Out	ϕ	track	
	β	v dot	
Steady Side Slip	ψ	track	β = target
	ϕ	v dot	
Other Rules	**State**	**Error**	**Constraints**
Ground Trim	h	vertical acceleration	δt = target
	θ	q dot	
	ϕ	p dot	
	ψ	track error	
Lateral Surfaces	δa	p dot	
	δr	r dot	
Pitch Rate	δe	q dot	

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37264

AIAA-2000-4390

MANAGING THE PLAYBACK AND RECORDING OF SIMULATION MODELS IN AN OBJECT-ORIENTED SIMULATION

P. Sean Kenney*

Systems Development Branch
NASA Langley Research Center
Mail Stop 125B
Hampton VA 23681

Paul C. Sugden†

Unisys Corporation
NASA Langley Research Center
Mail Stop 169
Hampton, VA 23681

Abstract

Flight simulation often uses playback as a means to replay the movement of vehicles in a simulation environment. Playback vehicles must be absolutely repeatable and computationally efficient. This paper presents an object-oriented design for playing back vehicle behavior using a direct vehicle-state playback, control-surface playback, or pilot input playback. The design was added to the Langley Standard Simulation in C++ (LaSRS++) where it supports a number of current experiments. The design provides an accurate means to replay the dynamic behavior of a vehicle while minimizing computational load. The design has proven to be flexible, maintainable, and extendible.

Introduction

Among the many things required to provide a repeatable flight environment, one of the more complex items is the presentation of other vehicles to a pilot. In military aircraft simulations a pilot may have to acquire and destroy drones, maneuvering aircraft, fixed-based ground targets, or moving ground targets. In commercial aircraft simulations a pilot may have to deal with air and ground traffic while flying or taxiing in an airport's vicinity. Qualitative research data can only be obtained if the vehicles in these scenarios move with absolute repeatability.

Another complex requirement is to provide the means to replay the flight profile of a vehicle. Flight data from an aircraft that has crashed can be analyzed by re-creating the flight. Simulation models can also be verified against test flight data. Analysis is difficult without a simple, repeatable method of injecting data into a simulation model.

---

* Aerospace Engineer, Member AIAA.

† Software Engineer

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

The Langley Standard Real-Time Simulation in C++ (LaSRS++) provides this repeatability using three different methods: state playback, control-surface playback, and pilot playback[1]. State playback records the states of a vehicle and uses the data to "playback" the vehicle's movement without performing any integration. Control-surface playback records the control surface commands or deflections and uses the data to recreate the dynamic movement of a vehicle by driving control surfaces with the commands or deflections and integrating the state of the vehicle. Pilot playback records the inputs made by a pilot and uses the data to playback the flight by injecting the recorded pilot commands into the fully operational simulation. Each of the three methods is applicable to different kinds of simulation research.

State playback is applicable whenever pilot interaction with another vehicle is only dependent upon that vehicle's position, velocity, and orientation. For example, a fighter pilot might be tasked to target and destroy another aircraft. To accomplish the task, the pilot must be able to see the CGI representation of the target, track the target with radar, and use a gun sight to destroy the target aircraft. The CGI model, radar model, and gun model only need the target's states to compute the its position, velocity and orientation relative to the pilot. Derived states for an aircraft like angle of attack, Mach number, and sideslip are not needed. Thus, state playback computational load is light when compared to a complete six degree of freedom simulation model. This allows multiple state-playback models to be used within the time constraints of a real-time simulation frame.

Control-surface playback supports analysis of the simulation that requires removal of control law influences. For example, surface commands or surface deflections can be obtained from flight test data to recreate the flight profile of the test vehicle. The performance of the simulation model can then be validated against the flight profile of the test vehicle. Control-surface playback isolates the performance of a model from its control system, thereby allowing a developer to measure the performance of modifications to the aerodynamic package. Moreover, the developer can test the aerodynamics package before the control law has been completed.

Pilot playback is used to analyze the response of a human to his/her environment or the response of a simulation model to a series of human inputs. For example, if a pilot is tasked to perform emergency maneuvers as part of a training exercise, the pilot's inputs can be replayed while the instructor discusses the pilot's performance. State-playback would not provide all of the derived information necessary to completely re-create the test sequence as viewed by the pilot. Another application of pilot-playback is to analyze any unexpected behavior produced by a simulation model. If a pilot encounters an unexpected event while flying, the recorded inputs can be used offline to repeat the scenario so that the problem may be analyzed without requiring all of the resources necessary to perform pilot-in-the-loop, synchronous real-time simulations. Pilot playback also provides an ideal means to generate a demonstration of a simulation facility without a pilot in the loop thus allowing visitors an unobstructed view of the facilities.

Design Requirements

An object-oriented design was created to allow the playback and recording of dynamic vehicle data in a simulation framework. The design had the following requirements:

1. The design must provide several general capabilities that may be incorporated into any type of playback.
   a. The design must allow a vehicle to specify its starting point within the playback file.
   b. The design should have a minimal impact on the performance of the framework when there are vehicles in playback mode and no impact on performance when there are no vehicles in playback mode.
   c. The design must provide a method to delay the start of playback for a specified amount of time.

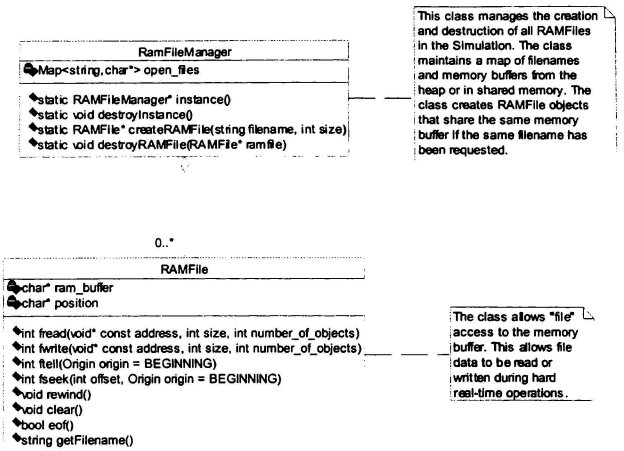


Figure 1 – RAMFile and RAMFileManager

d. The design must provide a means to continuously loop.
e. The design must provide a mechanism for recording data at integral divisions of the simulation frequency.
f. The design must provide a mechanism for playing back data recorded with a different frequency than the simulation.
g. The design must allow multiple vehicles to be driven from one playback file.

2. The design should provide several specific playback capabilities.
   a. The design should support an airport traffic mode where the playback vehicles have some intelligence when taxiing. A vehicle should not run into other vehicles, should not incur upon the runway when another aircraft is taking off, should stop smoothly when forced to halt, etc…
   b. The design should support the playback of surface deflections or surface commands.
   c. The design should support the playback of cockpit inputs.

A design that met the above requirements was implemented in the Langley Standard Real-Time Simulation in C++ (LaSRS++) application framework. LaSRS++ provides a powerful object-oriented framework for dynamic vehicle simulation in real-time[3]. The framework's object-oriented design makes the software extremely flexible, easily maintainable, and provides a high degree of re-use[4]. Encapsulation was used to hide implementation details. Inheritance and aggregation of common subcomponents were used to achieve maximal code reuse. Virtual method interfaces were utilized to obtain the advantages of polymorphism. Documented design patterns were used where possible[2]. The Builder and Singleton patterns were used in the design. The Builder pattern separates the construction of a

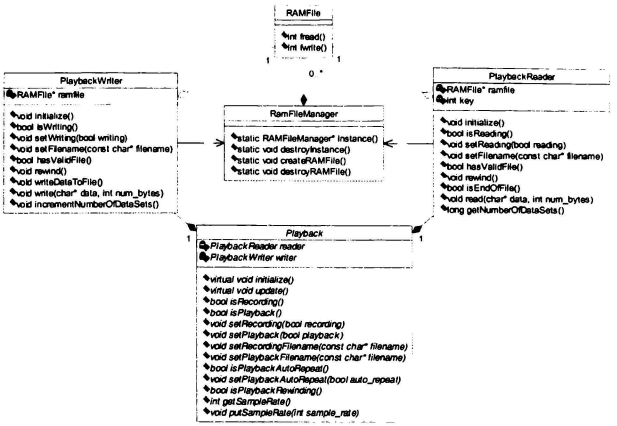


Figure 2 – Playback

complex object from its representation so that the same construction process can create different representations. The Singleton pattern ensures only one instance of a class exists and provides a global point of access to it.

RAMFile

The core of the design is a class called *RAMFile*. This class allocates a block of memory on the heap or in shared memory, and provides an interface that allows a client to treat the block of memory like a UNIX file. A client can read or write the "file" without the performance penalty associated with actual file access. This allows a client to write data to the *RAMFile* during hard real-time operations and then copy that data from the *RAMFile* to physical disk when the constraints of hard real-time are no longer in effect. The class also allows the file to be accessed by more than one client at a time. Figure 1 uses the Unified Modeling Language (UML) to show the class diagram for the *RAMFile* class and its relationship to a second class named *RAMFileManager*.

RAMFileManager manages the creation and destruction of all *RAMFile* objects in the simulation. The Singleton pattern ensures that only one instance of this class is ever created. *RAMFileManager* fields requests by clients to create *RAMFile* objects via the *createRAMFile* method. The method returns a *RAMFile* reference. If a client is the first client to request that the contents of a file be used in a *RAMFile*, *RAMFileManager* allocates a memory buffer and loads the file's contents into the buffer. The *RAMFileManager* then passes the buffer to the *RAMFile* when it is instantiated. If a client asks for a file that has already been loaded into a buffer for use by another client, the *RAMFileManager* class instantiates a *RAMFile* object by passing the pre-existing memory buffer to *RAMFile*'s constructor. This allows multiple clients to work on a single file without complicating the interface of *RAMFile*. Similarly, each client calls *destroyRAMFile* when the client is finished

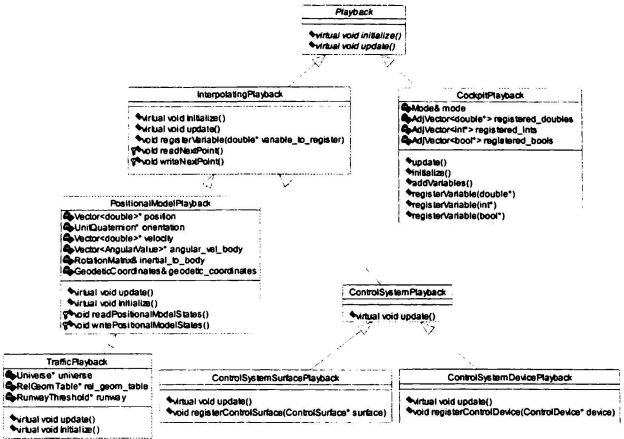


Figure 3 – The Playback Hierarchy

using a particular file. *RAMFileManager* only destroys the memory buffer for a file when there are no longer any *RAMFile* objects referencing the buffer.

Playback

Playback is an abstract class intended to be the base class for all types of playback operations. The class was designed to provide a generic interface for the recording and playback of data. Figure 2 demonstrates the Playback class and its dependencies.

Playback has two virtual methods that must be defined by all concrete derived classes. The *initialize* method is expected to initialize a derived class before or after data has been recorded or played back. Transfers to/from physical disk should occur when this method is called. The *update* method is intended to actually transfer data between the *RAMFile* and the simulation models. All of the other methods defined in Playback are to be used by clients and derived classes to manage the two classes contained within *Playback*, *PlaybackReader* and *PlaybackWriter*.

PlaybackReader provides an interface for reading playback data. If a valid filename has been given and the *setReading* method has been called with *true* as its argument, the class requests *RAMFileManager* to create a *RAMFile* for the file and then copies the data into the RAMFile if it is the first client of the *RAMFile*. If it is not the first client of this *RAMFile* then the data has already been loaded into memory from the file. Aggregation of a *RAMFile* object rather than inheritance allows *PlaybackReader* to hide the subset of *RAMFile*'s interface for modifying the file, while providing wrapper methods allowing reuse of *RAMFile*'s reading capabilities. The *PlaybackReader* class can now be used to read the data.

PlaybackWriter works in a similar manner except it writes data into the *RAMFile* and then copies it to physical file when *initialize* is called. The class assumes that it is the only user of a *RAMFile* because it is writing data. In a manner similar to *PlaybackReader*, *PlaybackWriter* prevents clients from accessing the reading capabilities of *RAMFile*'s interface

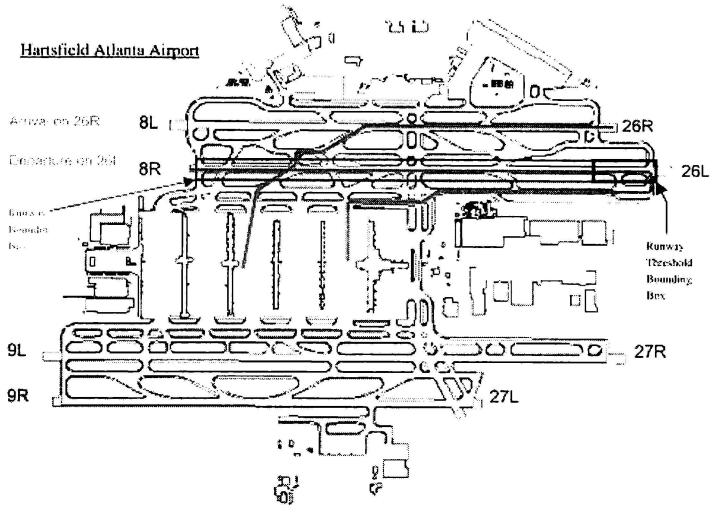


Figure 4 – Airport Traffic Diagram

while providing access to *RAMFile*'s writing capabilities through wrapper methods.

Interpolating Playback

InterpolatingPlayback is a concrete class that records data at a fixed integral division of the simulation frequency while allowing data to be played back every frame by interpolating between the sampled data points. Figure 3 illustrates that InterpolatingPlayback inherits from Playback and that the class implements the *initialize* and *update* virtual methods declared in the interface of *Playback*. The class also uses all of the methods defined in *Playback* to manage the recording and playback of files.

The method *registerVariable* allows client classes to register references to variables that are to be recorded and played back. Each variable that is registered with InterpolatingPlayback is added to a vector of variables. When recording, the method *writeNextPoint* first writes the current time to the *RAMFile* and then iterates through the vector of registered variables writing the value of each variable to the *RAMFile*. The *update* method makes sure that the *writeNextPoint* method is only called at the selected recording frequency. During playback, when the *update* method is called *readNextPoint* is called to read data sets from the playback file until it finds one which occurs after the current simulation time. The method *readNextPoint* saves this set and the previous one. A linear interpolation between these two sets determines the current set. These calculated values are then copied into the registered variables. If the recording frequency equals the playback frequency, no interpolation will be performed

and the played back data will be identical to the recorded data.

PositionalModelPlayback

In the LaSRS++ framework, a *PositionalModel* is a simulation object that has a position, orientation, translational velocity, angular velocity, and a set of geodetic coordinates. These are the minimum attributes required by the framework for a simulation object to be represented visually using a CGI and to support the computation of relative geometry between simulation objects. The *Vehicle* class extends *PositionalModel* by inheriting from it and adding acceleration as a state. *Aircraft* in turn, extends *Vehicle* by adding all of the states of an aircraft (angle of attack, sideslip, etc...). Since all vehicles in the LaSRS++ framework are descendants of *PositionalModel*, the motion of any vehicle may be recorded and played back by recording the states of a *PositionalModel*. The *PositionalModelPlayback* class provides a means to record and playback these states.

The constructor of *PositionalModelPlayback* requires references to the state vectors of *PositionalModel*. This is possible because *PositionalModelPlayback* is contained within *PositionalModel*. *PositionalModelPlayback* inherits from *InterpolatingPlayback* and uses the *registerVariable* method to register each component of the states. When the *update* method is called, the class uses the *update* method in *InterpolatingPlayback* to record or playback the state data. If playing back data, the method then computes several derived quantities needed by the simulation framework.

TrafficPlayback

TrafficPlayback is a class that extends *PositionalModelPlayback* by providing logic to make a playback positional model behave like aircraft traffic around an airport. The logic assumes that all traffic is centered about a pair of parallel runways and that the arriving traffic must cross the departure runway. The logic also assumes that the behavior of a recorded aircraft is consistent with that of an aircraft operating at an airport. Figure 4 shows a typical airport layout where the *TrafficPlayback* class can be used. On this diagram all departing aircraft are leaving on runway 26L and all arriving aircraft are on 26R. The arriving aircraft must cross the departure runway to get to the terminal.

The class performs several different computations to ensure that a playback behaves like an aircraft while taxiing. First, it uses the relative geometry information computed by the framework to ensure that the model is not going to collide with another model. If a collision is predicted, then the simulation model is brought to a halt until the path is clear again. Next, the class manages whether a model may begin to cross or enter the departure runway. To do this the class preprocesses the playback file when the class is first constructed and then determines on which frames the playback model would enter or exit the departure runway. The departure runway is broken into two zones: the runway threshold bounding box and the runway bounding box as illustrated on figure 4. By tracking when a model is entering either of the two zones, the class can manage whether or not the model can enter either area because it also knows whether another model is active in either part of the runway. For example, if a departing aircraft has already moved onto the threshold and is ready for takeoff, any arriving aircraft that is already crossing the departure runway on its way to the terminal would be allowed to continue. The departure aircraft is not allowed to begin its takeoff until after the runway is clear. But any arriving aircraft that are approaching the departure runway are forced to stop at the hold short line (the outer edge of each bounding box) so that the departing aircraft can take off. Once the departing aircraft is airborne the arriving aircraft is allowed to continue.

Finally the class gives the appearance of a continuously busy airport by rewinding a file when the playback has reached its end and then starting over. A departure playback file, for example, starts at a terminal and taxis to the departure runway. After takeoff the aircraft flies for several more minutes. By repeating the file again, one aircraft can be used over and over to give the appearance of a busy airport. The class also allows the simulation user to specify a starting location in the playback file by specifying a latitude, longitude and altitude. This allows multiple models to use the

same playback file but start at different time locations within the file. The class also allows the simulation user to specify the amount of time the model should pause before starting playback. This option allows the simulation operator to subject a pilot to traffic events such as near collisions, runway incursions, and other unusual circumstances.

ControlSystemPlayback

In the LaSRS++ framework, a *ControlSystem* maintains a collection of *ControlSurfaces*. A *ControlSurface* has a position command and a deflection. In closed loop simulation, control laws transform pilot inputs into surface position commands. The particular control surface then transforms the commanded position into a deflection.

ControlSystemPlayback is derived from *InterpolatingPlayback* and serves as a base for recording and playback of data required for repeatability during playback of control surface commands or deflections. Since the behavior of an aircraft is dependent both on the control surfaces and the engine thrust, it is necessary to record and play back the throttles so that playback of the control system results in a model with identical behavior to the recorded model. *ControlSystemPlayback* overrides the virtual *update* method of *InterpolatingPlayback* to transparently record and playback throttle positions. It is left as the responsibility of child classes to record and playback surface commands or positions.

The relationship between *ControlSystem* and *ControlSystemPlayback* is analogous to that between *PositionalModel* and *PositionalModelPlayback*. *ControlSystemPlayback* objects are constructed by and contained within *ControlSystem* objects. It is the responsibility of the *ControlSystem* object to expose its internal state to the playback object. The mechanisms by which the *ControlSystem* exposes its state are defined in the two child classes of *ControlSystemPlayback*. *ControlSystemDevicePlayback* defines a method *registerControlDevice* that gives the playback a reference to one of the *ControlDevice* objects contained within the *ControlSystem*. Similarly, *ControlSystemSurfacePlayback* defines *registerControlSurface* to allow a *ControlSystem* to expose a *ControlSurface* reference to a playback object.

ControlSystemDevicePlayback maintains a mapping of *ControlDevices* to commands. For each device registered with a *ControlSystemDevicePlayback*, an entry is added to this mapping, and the command is registered with the ancestor class *InterpolatingPlayback*. The virtual method *update* is overridden in *ControlSystemDevicePlayback*. In playback mode, the *update* method of the parent class is called to drive the throttles and to obtain new values for the device commands. In recording mode, the *update* method records the throttle positions and the device commands at the recording sample rate. *ControlSystemSurfacePlayback* operates in a similar fashion, for the purpose of recording and playback of throttles and surface positions.

CockpitPlayback

CockpitPlayback provides a means to record the pilot's inputs during a simulation. In LaSRS++, aircraft are currently regimented into three categories: fighter, transport, and drop model. Each aircraft type has a specific cockpit interface that provides the aircraft with its inputs. The *CockpitPlayback* class is used by these three interfaces to record and playback a pilot's inputs.

A particular cockpit interface registers the variables that are to be recorded and played back just like *InterpolatingPlayback*. *CockpitPlayback* however does not have a sampling rate – it records data every frame. This is the only way to ensure that the dynamic response of a model during playback is identical to the original performance.

Summary

The design presented in this paper met all of the specified requirements. The functionality of *TrafficPlayback* alone demonstrates that the design is flexible enough to provide most of the required general capabilities. It allows a vehicle to specify a starting point within a playback file. It allows the simulation operator to start playback at any time. It repeatedly uses the same playback data to affect the motion of a positional model. It allows several vehicles to be driven from the same data file. *InterpolatingPlayback* demonstrates

fulfillment of the remaining general requirements. It provides a means of recording data at integral divisions of the simulation frequency. It provides a means of playing back data that was recorded at any frequency. The specific playback capabilities were fulfilled by the concrete, derived classes.

Each of the playback methods has been used in a number of experiments at NASA Langley Research Center (LaRC). For example, the *TrafficPlayback* class was used in the Low Visibility and Surface Operations (LVLASO) study in the Research Flight Deck (RFD) at LaRC. The project used the B757 model to study a means of improving the safety and efficiency of airport surface operations so that clear-weather capacities can be maintained during instrument weather conditions. The TrafficPlayback class was used to provide airport traffic during the study. The *TrafficPlayback* proved to be extremely efficient. When the B757 aircraft model is used by itself in the RFD it normally consumes on average 11 milliseconds of CPU time per frame. When traffic was added to the simulation using sixteen instances of the *TrafficPlayback* class, the average cpu time consumed per frame increased to only 13 milliseconds. Clearly each traffic model added to the simulation adds very little to the computational requirements of the framework. The *ControlSystemPlayback* class has also been used in several projects. For example, the Weather Accident Prevention (WxAP) project played back surface positions from flight test data to verify simulation model performance. The *CockpitPlayback* class is used on a daily basis by both developers and researchers to analyze the performance of both pilots and simulation models.

Object-oriented development in C++ allows the complexities associated with playback to be abstracted away behind concise interface definitions of the general purpose classes discussed in the present work. The diverse uses of *Playback* and *InterpolatingPlayaback* suggest that additional special purpose recording and playback features can be easily added to the framework. Development and maintenance efforts have been minimized by reuse of this robust foundation.

Although the design presented in this paper was originally designed to support flight simulation at NASA Langley Research Center, the design could be used in any object-oriented framework to heighten reuse and maintainability.

Bibliography

[1] Richard A. Leslie, et al. *LaSRS++ An Object-Oriented Framework for Real-Time Simulation of Aircraft*. Paper Number AIAA-98-4529, August, 1998.

[2] Gamma E., Helm R., Johnson R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, Massachusetts, 1995.

[3] Michael Madden, et al. *Constructing a Multiple-Vehicle, Multiple-CPU Using Object-Oriented C++*. Paper Number AIAA-98-4530, August, 1998.

[4] David Geyer, et al. *Managing Memory Spaces In An Object-Oriented Real-Time Simulation*, Paper Number AIAA-98-4532, August, 1998.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37265

AIAA-2000-4391

AN OBJECT-ORIENTED INTERFACE FOR SIMULINK MODELS

Michael M. Madden*

NASA Langley Research Center
MS 125B
Hampton, VA 23681

Abstract

Visual programs for modeling dynamic systems have become popular because they simplify the construction and maintenance of math models. These products also offer an appealing variety of analytical tools that aid the evaluation of model performance. With the addition of code generators, these tools can convert their visual diagrams into source code that can run standalone or be integrated within an existing software product. The simulation industry has experienced a growing trend of integrating generated code with existing, simulation frameworks. Because the simulation framework and the auto-code generator are based on independent designs, developers must add software layers that overcome incompatibilities. Ideally, the developer strives to design a reusable interface for the visual-modeling product. This frees future projects from handling auto-code integration issues. However, achieving a reusable interface may require customization of the visual modeling product and restrictive guidelines on its use.

Many simulation customers at NASA Langley Research Center have selected Mathworks' Simulink® for visual modeling. Mathworks supplies a code generator for Simulink called the Real-Time Workshop® (RTW). The simulation software group at Langley created a reusable, object-oriented interface that integrates RTW code with the Langley Standard Real-Time Simulation in C++ (LaSRS++). LaSRS++ is an object-oriented framework for creating the real-time simulations that run in Langley's simulator facilities.[1]

Introduction

The design of the LaSRS++/RTW interface has two main goals. First, LaSRS++ should treat all vehicles equally, whether or not they contain RTW code. Second, the design should avoid changes to the auto-code and to the user's process for generating the auto-code. The latter protects the interface against obsolescence with future RTW releases and minimizes additional procedures for the Simulink user. These goals result in the following requirements:

1. The interface shall support execution of auto-code in the simulation modes: RESET (initialization and scenario definition), TRIM (solve for steady state at initial condition), HOLD (freeze the simulation), and OPERATE (propagate through time).
2. The interface shall support the execution of auto-code through multiple cycles of RESET-TRIM-HOLD-OPERATE. In other words, a simulation with auto-code must be able to run several scenarios or the same scenario repeatedly without the need to shutdown and re-execute the simulation.
3. The interface shall allow multiple, heterogeneous Simulink models and multiple copies of the same model to operate in the same simulation. LaSRS++ supports multiple, heterogeneous vehicles in a simulation. Failure to support multiple, heterogeneous Simulink models limits vehicle models that contain auto-code to a single instance

* Senior Member, AIAA

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty free license to exercise all rights under the copyright claimed herein for Government Purposes. All other rights are reserved by the copyright owner.

in a simulation. It also prevents a vehicle model from containing more than one auto-coded Simulink model. Failure to support multiple copies of the same model precludes reuse of auto-code within a simulation. For example, a user could not build an actuator model in Simulink, generate auto-code from it, and use an instance of the auto-code for each control surface.

4. The interface shall use the auto-code as generated. To add new features to the auto-code, developers must customize RTW code generation by modifying the Target Language Compiler (TLC) scripts.

Applying these basic requirements reveals a host of problems. Many are rooted in the design of RTW auto-code, which focuses on real-world execution rather than execution in a simulation. The biggest issues are:

1. RTW does not support a "trim" mode; the auto-code initializes to a defined state then operates.
2. RTW auto-code is designed to run once, i.e. in a single initialize-operate loop-terminate cycle.
3. Until version 3.0, RTW did not directly support multiple models or copies of the same model in the same program. This support is provided in the newly introduced Generic Real-Time Malloc (GRTM) Target. Thus, the LaSRS++/RTW interface will not work with prior versions of RTW.

In addition, the RTW auto-code design conflicts with two major LaSRS++ design goals:

1. Minimizing transport delay[†] is a design goal for LaSRS++. RTW's design for resolving continuous states tends to increase transport delay.
2. LaSRS++ simulations are designed to run at any fixed frame rate. The frame rate for a simulation is established at runtime. RTW generates auto-code with a pre-determined frame rate that is taken from the simulation parameters defined within Simulink.

All of the above issues are overcome with a combination of interface design choices, user guidelines on Simulink model construction, modification of TLC files, and procedural workarounds.

[†] Transport delay is time between the generation of input and its visible affect on vehicle dynamics. Frequently, the term is used to refer to the time between pilot input and the response of the visual or motion system to that input.

Overview of the RTW code design

RTW uses the Target Language Compiler (TLC) to generate code from a model. The code-generation rules are packaged in a set of scripts processed by the TLC. A package of TLC scripts is called a *target*. RTW ships with ready-to-use targets for a variety of runtime environments. The Generic Real-Time Malloc (GRTM) target that is required for the LaSRS++/RTW interface follows the basic design described below.

All model data is connected directly or indirectly to a central structure of type Simstruct. Access to the model data is coordinated through a Simstruct instance, of which there is only one instance per model. Because the Simstruct type is subject to change in future versions, the Simstruct instance is not intended for direct access. Instead, RTW supplies functions for manipulating the Simstruct data. In this manner, it encourages object-like interaction with the Simstruct data.

The generated code consists of three classes of functions: initialization, operation, and termination. The initialization functions include the model registration function which is named after the model and the MdlStart() function. The model registration function constructs, populates and initializes the Simstruct instance. The MdlStart() function initializes all states in the model and performs other one-time initialization tasks. The operation functions include MdlOutputs(), MdlUpdate(), and MdlDerivatives(). MdlOutputs() computes the output of each block in the model. MdlUpdate() updates discrete states. MdlDerivatives() calculates the derivatives for continuous states in each block. The termination function is called MdlTerminate(); it executes shutdown code for each block.

RTW includes a code library that is reusable across all models. This library provides fixed-step integration functions, a data logging function, and external mode[‡] interface functions for the generated code.

[‡] External mode is a feature that allows the Simulink block diagram to communicate with an external process executing the RTW code.[6]

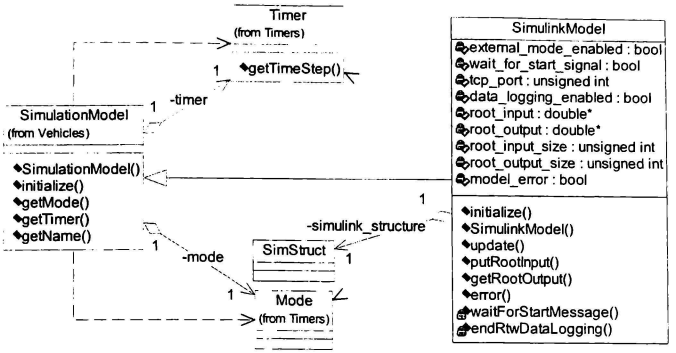


Figure 1 UML Class Diagram of Simulink-RTW Interface

LaSRS++ Interface Design

The LaSRS++/RTW interface is encapsulated in a single class, SimulinkModel. Figure 1 illustrates the design of SimulinkModel in Unified Modeling Language (UML) notation.[2] Figure 2 displays a simplified representation of the SimulinkModel header file. SimulinkModel is a C++ class wrapper for the auto-code. The class encapsulates the details of operating the auto-code. Direct access to the Simstruct instance and RTW functions is not provided. Instead, the class replaces the complexity of operating the auto-code with a very simple interface.

SimulinkModel presents the auto-code as a system with single-dimensioned input and output vectors. Inputs are defined using the putRootInput() method. Outputs are extracted using the getRootOutput() method. Both methods use an integer index argument to identify the data being manipulated. The SimulinkModel defines just four behaviors for the model: construction, destruction, initialize(), and update(). The constructor builds and initializes the Simstruct instance using the model's registration function. The constructor also performs one-time initializations on the model. The registration function is an argument to the constructor. Each SimulinkModel object can represent a unique Simulink model. The registration function passed to the constructor determines which model a given SimulinkModel object will contain. The initialize() method resets the model to an elapsed time of zero. The simulation calls initialize() in RESET mode. The update() method operates the model; as will be discussed later, it is called in both TRIM and OPERATE modes of the simulation. The destructor frees all of the memory used by the auto-code and terminates its operation. SimulinkModel also provides an error monitoring method, error(). This method returns true if the auto-code reports a non-fatal error since the last time error() was called. SimulinkModel handles all error reporting. The client code decides whether it will continue past a non-fatal error.

SimulinkModel derives from SimulationModel. SimulationModel is the base class for all math models in LaSRS++. Thus, LaSRS++ treats the SimulinkModel as a specialized math model. SimulinkModel must support the basic behaviors of all SimulationModels. SimulationModels contain references to a Timer, which encapsulates the time step for the SimulationModel, and to a Mode, which contains the current simulation mode. SimulationModels must behave appropriately based on the information in Timer and Mode.

```
#include "SimulationModel.hpp"

extern "C" {
#if !defined(RT) // Mathwork's simstruc.h requires that RT be defined for RTW code.
#define RT
#endif
#include "simstruc.h"  // SimStruct cannot be forward declared; it is a typedef.
typedef SimStruct* (*SimulinkModelConstructor)(void);  // Model registration function.
}

class SimulinkModel: public SimulationModel {
public:
  SimulinkModel(const Mode&              simulation_mode,        // Simulation mode.
                const Timer&             simulation_timer,       // Time step for model.
                SimulinkModelConstructor model_constructor,      // Registration function.
                bool                     use_external_mode,      // Enable external mode.
                unsigned int             external_mode_tcp_port, // Port # for external mode.
                bool                     wait_for_start_message, // External mode option.
                bool                     data_logging);          // Enable data logging.

  virtual ~SimulinkModel();
  virtual void initialize();   // Redefine SimulationModel::initialize().
  virtual void update();       // Operate the Simulink model.
  void putRootInput(double input, unsigned int index) (root_input[index] = input; );
  double getRootOutput(unsigned int index) const      (return root_output[index]; );
  bool error() const {
    bool return_error = model_error; model_error = false; return return_error; };

private:
  SimStruct*   simulink_structure; // Model data and pointers to model functions.
  mutable bool model_error;        // Set to true when model reports an error.

  // External mode and data logging option variables.
  bool         external_mode_enabled; // Enables external mode if true.
  bool         wait_for_start_signal; // Sim pauses for simulink start message.
  unsigned int tcp_port;      // TCP port number of external mode connection.
  bool         data_logging_enabled;  // Enables RTW data logging if true.

  // The simulink model I/O is represented as single dimensional arrays.
  double* root_input;
  double* root_output;
  unsigned int root_input_size;
  unsigned int root_output_size;
};
```

Figure 2 Simplified Header File for SimulinkModel

Implementation of the SimulinkModel class borrows heavily from the grt_malloc_main.c file that ships with RTW[9]. As a result, SimulinkModel can call RTW library functions for data logging and external mode[1]. Both features are optional. Arguments to the SimulinkModel constructor enable and configure these options. RTW data logging can be enabled for more than one model in the same program. Only one SimulinkModel object can enable external mode because the external mode code uses global data.

RTW data logging allows the customer to select, through the Simulink product, variables internal to the Simulink model for recording. However, since RTW code is designed to run in a single initialize-operate-terminate cycle, the RTW data logging is only guaranteed to function for the first run of the simulation. The data logging functions may return errors on subsequent runs. These errors are not fatal and do not affect the operation of the model, but the resulting data log may corrupt. The presence of ToWorkspace blocks in a Simulink model require that data logging be turned on because RTW adds their input signals to the data log. Disabling data logging for a model with ToWorkspace blocks causes a segmentation violation during operation. By default, RTW data logging bases its memory requirements on the start and end times defined for the

[9] RTW supplies this file for creating a standalone executable from the GRTM generated code.

[1] External mode operation has not been tested to date.

model. The SimulinkModel modifies the Simstruct so that it will run for infinite time[#]. Obviously, RTW cannot allocate memory for an infinite-time run. RTW will issue a warning and use a default buffer size of 1024. To remove the warning and use a different buffer size, the Simulink model developer must set **Simulation -> Parameters -> Workspace I/O -> "limit rows to last"**, prior to generating code.

Though it significantly reduces the complexity of adding RTW code to LaSRS++, SimulinkModel is not a complete solution. The SimulinkModel class does not know how the indices of its input and output vectors map to other variables in the LaSRS++ simulation. The software engineer must still create client code that directs input data from the LaSRS++ simulation into SimulinkModel and directs output data from SimulinkModel into the simulation. The client code must also monitor SimulinkModel for non-fatal errors and trigger appropriate actions. The class must also exercise the update() method during the appropriate LaSRS++ modes as discussed in the section "Simulation Mode Support".

Solving Interface Issues

The simplicity of the LaSRS++/RTW interface design masks the implementation choices and procedural workarounds necessary to meet the requirements stated in the introduction. This section details these solutions.

Multiple Simulink Model Support

Only one target supports multiple Simulink models in the same program, the Generic Real-Time Malloc (GRTM) target. The LaSRS++/RTW interface will operate only with auto-code generated by the GRTM target. GRTM first appeared in RTW 3.0. Thus, the LaSRS++/RTW interface requires RTW 3.0 or later. Furthermore, the LaSRS++/RTW interface only works with code generated for single-task operation.

GRTM differs from the other pre-packaged targets in two fundamental ways to enable multi-model support, as illustrated by comparison against its older sibling, the generic real-time (GRT) target. First, GRT statically allocates all persistent data with a mixture of internal and external linkage; moreover, GRT uses the same variable names for the major data structures in all auto-code that it generates. This prevents multiple Simulink models from running in a program. For each common name with external linkage, the linker would assign the same memory location. All Simulink models in a program would then share the same memory locations, leading to data corruption. On the other hand, GRTM dynamically allocates all data structures and effectively binds them to a Simstruct pointer returned by the model's registration function. Thus, each instance of a model receives its own copy of persistent data, and the generated code contains no common variable name with external linkage.

Second, GRT uses the same names for functions that operate the model[**] and gives these functions external linkage. Thus, distinct Simulink models cannot exist in the same program because the function names would conflict at link time. GRTM avoids this conflict by declaring the functions with internal linkage. Only the registration function has external linkage and its name is the same as the model. Thus, all distinct models must have unique names to coexist in the same program. To make the operation functions available externally, GRTM places function pointers in the Simstruct structure and assigns the addresses of the generated functions to the pointers. The functions for a Simstruct instance are executed by exercising the function pointers that it contains; i.e., each Simstruct structure is bound to the functions that act on it. This technique is similar to the virtual function mechanism in C++.[3]

Unfortunately, the integration algorithm is not among the functions stored as a pointer in the Simstruct instance. Thus, all models in a program must use the same integration algorithm. In fact, the Simulink user's selection of integration algorithm in the Simulation Parameters dialog has no meaning to the LaSRS++/RTW interface. The setting does not influence the generated code. It only influences the tem-

[#] This is accomplished by calling ssSetTFinal() to set the final time to zero. Simulink interprets a final time of zero as equivalent to an infinite final time.

[**] These functions are MdlStart(), MdlOutput(), MdlUpdate(), MdlDerivatives(), and MdlTerminate().

plate makefile that RTW generates with the code; it adds the file containing the selected algorithm to the list of files to be compiled and linked. LaSRS++ simulations use their own makefiles and ignore the generated makefile. The LaSRS++ makefiles allow the program creator to select any of the RTW integration algorithms at build time. The simulation will apply the algorithm, with which it is linked.

The library code shipped with RTW is reused across many targets. This code uses pre-processor variables to customize itself for a specific target. The variable RT must be defined when compiling the code for all RTW targets. The variable RT_MALLOC must be defined for the GRTM target. Thus, the LaSRS++/RTW interface and all RTW library code must be compiled with the variables RT and RT_MALLOC defined. In addition, on UNIX systems, the variable UNIX must be defined. Unfortunately, the RTW library code relies too heavily on pre-processor variables to also define the number of continuous states in a model (NCSTATES) and the number of sample times in a model (NUMST). Obviously, code utilizing these pre-processor variables will be compiled with only one value for each variable and those values will be applicable to only one model. All other models would not function properly, defeating the ability to have multiple models in the same program. However, the Simstruct structure contains the data represent by these variables; and functions[††] are supplied to extract this data. NCSTATES can be replaced with a call to ssGetNumContStates(). The function ssGetNumSampleTime() can be substituted for NUMST. By examination of the RTW library source, it is apparent that some attempt had been made to remove these pre-processor variables from code exposed when RT_MALLOC is defined; however, the task was left incomplete. The integration algorithms still contain a reference to NCSTATES. To repair this defect, a copy of the code was made for the LaSRS++/RTW interface and NCSTATES was replaced with ssGetNumContStates(). Likewise, the code for external mode references NUMST. Once again, a copy of the code was made; and NUMST was replaced with a call to ssGetNumSampleTime()[‡‡]. Also, grt_malloc_main.c uses NCSTATES and NUMST heavily. When sections of this code were copied into the methods of SimulinkModel, NCSTATES and NUMST were replaced with their functional equivalents.

Transport Delay

The LaSRS++ simulation framework minimizes transport delay as a design goal. As one means of achieving this goal, LaSRS++ simulations resolve continuous states (i.e., integrates derivatives) as they occur in the execution path, and LaSRS++ propagates the results immediately. Simulink, on the other hand, collects the computed derivatives and simultaneously integrates the derivatives as the final step of operation. At best, the output of each integrator will not affect the behavior of the vehicle until the following frame. If several states are chained together in the execution path, the inputs influencing the first derivative may not affect vehicle behavior for several frames. For single-pass integration algorithms, the input's influence on the model's output will be delayed one frame for each state on the chain. Simulations commonly use single-pass integration algorithms because they are computationally efficient; they are susceptible to this delay. Systems that use a large number of transfer functions are also vulnerable to this problem. Mathworks' design is correct for a continuous model. But, its discrete representation introduces an artifact in the form of transport delay. At the frame rates normally run for a simulation, the resulting delay can lead to vehicle response that the pilot finds unrealistic or that causes the pilot to adapt differently than in the actual aircraft.

The solutions are limited. One can write a new TLC that reflects the modeling philosophy of LaSRS++, but this solution requires a large resource commitment. The next two solutions require additional computation, which the computer may not be able to support. One could use a multi-pass integration algorithm. For a model with n states in its longest chain, the algorithm must run n passes. The input will now influence the last

[††] Most Simstruct functions are actually implemented as pre-processor macros.

[‡‡] A large number of function signatures in the external mode code also had to be modified to make the switch.

state in the chain with a single integration. But, the results are still delayed one frame. Also, a multi-pass algorithm mixes passes where the input does not influence the last state with passes where the input is influential. Passes where the input is influential remain until the nth frame after introduction. Thus, the input's total influence is spread over n frames. Alternatively, one can run the RTW code at a higher rate; ideally, the rate is the product of the simulation rate and n. This approach resolves the total influence of the input in one simulation frame. Depending on the purpose of the Simulink model, the vehicle plant[§§] may have to run at the higher rate to maintain realistic behavior. Thus, this approach is frequently more computationally expensive than the multi-pass integration algorithm. The processing power of the host computer may prevent the RTW code from running at the "ideal" rate, forcing the project to experiment with different simulation rates to balance computing resources and vehicle response.

Using discrete blocks instead of continuous blocks may not solve the problem. Like continuous states, discrete states are not updated by calling MdlUpdate() until after the block outputs are calculated by calling MdlOutput(). Some algorithms, like the Euler algorithms in the discrete integrator, produce pure delays. However, some discrete algorithms are spread over both the MdlUpdate() and MdlOutput() functions to take advantage of currently computed inputs, eliminating the delay. But, as discussed in the section "Simulation Mode Support", some of these blocks may not behave correctly for TRIM mode and cannot be used. Thus, the Simulink user must be aware of the limitations of each discrete block to effectively use them for simulation.

Variable Time Step

LaSRS++ allows the user to define the time step during simulation start-up. The Timer member in the SimulationModel base class communicates the simulation time step (or a multiple or integral divisor of the simulation time step for multi-rate simulations) to the math model. Hand-coded models are designed for a variable time-step. It is necessary for components that are reused with a variety of simulations. However, the default RTW code generator does not provide an interface for changing the time step of the auto-code. When the code is generated, RTW imposes the time step defined for the model in Simulink's "Simulation Parameters" dialog. The TLC could be modified to provide a means of changing the time step. But, the need does not yet justify the effort at LaRC since a workaround exists. When a new time step is required, the RTW auto-code is regenerated with the new time step, compiled, and linked into the simulation. Although the simulation projects at LaRC use a variety of time steps, an individual simulation project tends to settle on one time-step. Thus, the workaround, though laborious, is infrequently used. To protect against accidental mismatch of time steps in the auto-code and LaSRS++ simulation, the SimulinkModel class compares the time step that it inherits from Simulation model with the time step stored in the auto-code. If the time steps differ, SimulinkModel will issue a warning to the user.

Simulation Mode Support

RTW does not support the simulation modes in LaSRS++. To operate within a LaSRS++ simulation, the auto-code must behave properly in the four major modes RESET, TRIM, HOLD, and OPERATE. Furthermore, the auto-code must behave properly through multiple runs (i.e., cycles of RESET-TRIM-HOLD-OPERATE) without the need to shutdown the simulation and restart it.

In RESET, the simulation resets the elapsed time to zero and defines the scenario for the next run. During RESET, the simulation calls the initialize() method of all SimulationModel objects. SimulinkModel, which derives from SimulationModel, resets the elapsed time in the auto-code by calling sfcnInitializeSampleTimes() in its initialize() method. Any signals in the model that require initialization must be connected to a root-level "inport" block. The SimulinkModel client is responsible for passing the initial value to the SimulinkModel.

[§§] The vehicle plant models the physical aspects of the vehicle. Usually, it consists of the aerodynamics model, the engine model, the gear model, and the control surface models.

In aircraft simulation, it is undesirable to have transient behavior in the first few seconds of operation; in fact, such transients can prevent safe use of a motion system. The aircraft simulation must be initialized to a steady state before operating. The simulation contains algorithms that solve for the steady-state values of the simulation states. In LaSRS++, these algorithms are exercised in TRIM mode. LaSRS++ uses a simple trim algorithm of the form:[4]

$$x_{n+1}^i = x_n^i + g_n^i y_n^i \quad (i = 1, m)$$

where x is an independent state, g is a gain, y is a dependent value, and m is the number of equations required to trim the aircraft for the given scenario. The equation calculates a new state with the purpose of reducing the dependent values to zero; thus, the dependent values are frequently referred to as the error. The gains set the proper direction and scale to make the algorithm work. The algorithm operates in a loop. For each pass, the simulation operates the math models with the new states; then, the algorithm updates the states. The process repeats until the magnitude of the error vector becomes smaller than a defined tolerance.

To successfully and efficiently "trim" in LaSRS++, all math models modify their behavior in two fundamental ways. First, transfer functions calculate the steady-state output for their input in one pass[¶] and initialize their states to the steady-state condition. Second, integrators do not integrate. Their output is set to a value, possibly controlled by the trim algorithm. In many cases, the integrator's input (i.e., the derivative) must be reduced to zero; i.e., the derivative becomes an error value in the trim algorithm. It is not unusual for the derivative's corresponding trim state (i.e. the state with the greatest influence on its value) to be the integrator output.

The RTW auto-code has no feature similar to the TRIM mode in LaSRS++. TRIM behaviors must be designed into the Simulink model. The Simulink user must assign a root inport block to be the TRIM flag.[5] Currently, the SimulinkModel design does not force the first inport block to be the TRIM flag. Since SimulinkModel cannot know which signal is the TRIM flag, the LaSRS++ client code must set the flag appropriately. When the flag is high, the Simulink model will activate TRIM mode behaviors; otherwise, the model will behave normally.

All integrators in the model must be reset integrators.[5] If the integrator output is a state in the trim algorithm, then the initial condition signal must be connected to a root-level inport block. If the derivative is an error in the trim calculation, the input signal to the integrator must be connected to a root outport block. The reset switch must be connected to the TRIM signal. The reset integrator will work as expected if it outputs the reset input when the TRIM signal is high. This was the reset integrator's behavior in the Simulink 1.0. However, Mathwork's decided to change the behavior starting with Simulink 2.0. Now, reset integrators use the reset input when the reset switch changes value. The TLC for the GRTM target was changed to restore the original behavior of the reset integrator.

Transfer function blocks (discrete or continuous) cannot be used; the blocks have no ability to reset themselves or compute their steady-state output in one pass.[5] Instead, transfer functions must be re-implemented as a system block containing reset integrators, and they must be designed to return the steady state output when the elapsed time is zero. The research community at LaRC has created a library of these transfer function replacements. Other behavior modifications for TRIM mode must be linked to a switch whose decision input is the TRIM signal.

In TRIM, the client code is responsible for communicating states and errors to the trim algorithm. The client code also retrieves newly calculated states from the trim algorithm and communicates them to the SimulinkModel object. The client sets the TRIM input high and calls SimulinkModel::update() to compute new error values from the new states. SimulinkModel::update() has logic that prevents integration, incrementing time, and data logging while not in OPERATE mode. Integration and incrementing time would interfere with the TRIM algorithm. Data log-

[¶] In other words, the transfer function must supply the output that results if the input is applied for an infinite amount of time.

ging is undesirable in TRIM; the purpose of data logging is to record the model's behavior during operation. SimulinkModel provides these behavior modifications for TRIM mode without the need for intervention from the client code.

In HOLD, the simulation does not update its states; and the simulation time does not increment. The vehicle is frozen. This is best achieved by not exercising the math models. However, there are special cases where a math model needs to run in HOLD. Thus, SimulinkModel allows its update() method to be called in HOLD mode. However, as stated previously, SimulinkModel will not perform integrations, increment time for the Simulink model, or log data except in OPERATE mode. These operations would violate the intent of HOLD mode.

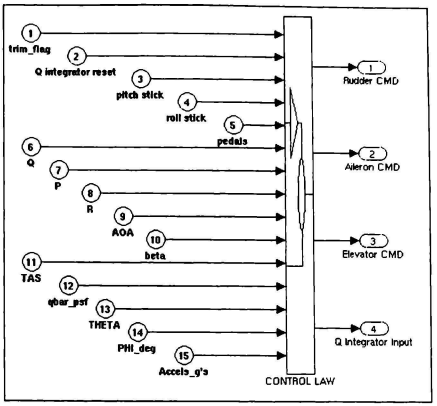


Figure 3 Example Simulink Model

In OPERATE, the client communicates inputs to the Simulink model by calling putRootInput(). Then, it calls update() for the SimulinkModel object. When update returns, the client extracts the SimulinkModel's outputs by calling getRootOuput() and communicates the outputs to the simulation. Optionally, the client will also call error() and take appropriate steps if an error is reported. The Simulink model requires no special considerations to behave properly in OPERATE; it is designed to operate.

The root issue that prevents RTW auto-code from performing correctly through multiple runs is that RTW does not provide an interface for resetting the model once it begins operation. However, the changes that enable correct behavior in RESET and TRIM mode also establish this missing interface. A Simulink model built using the guidelines above will safely restart without the need to completely terminate it and recreate it.

Example

This section provides an example of adding the SimulinkModel class to a LaSRS++ simulation. A customer has built a Simulink model called simple_control_law illustrated in Figure 3. This model contains an integrator in the longitudinal law that must be added to the trim algorithm. The derivative is the error and its output is the derivative's companion state. As required for the LaSRS++/RTW interface, the integrator's reset value is attached to inport block #2 and the derivative is exported through outport block #4. Also, a trim flag is installed as inport block #1. Though not visible, all transfer functions have been re-implemented with reset integrators. To illustrate how the root-level "port" blocks map to SimulinkModel's I/O vectors, the "Accels_g" block has a length of three.

Code Generation

To generate code, the Simulink user selects the Generic Real-Time Target and configures it for "code generation only" as specified in the *Realtime Workshop User's Guide*.[6] In this example, the user does not identify data for RTW data logging and the model does not contain ToWorkspace blocks. Thus, the user does not need to change settings for data logging. The user launches the Simulation Parameters dialog (**Simulation->Parameters->Solver**) and sets the solver to fixed-step and single-tasking mode. The user also sets the step size to the simulation step size. The user then generates the code by selecting **Tools->RTW Build**.

```
#include "SimulinkModel.hpp"

class Aircraft;  // Base class for all aircraft models.

class ExampleModel: public SimulinkModel {
public:
  enum InputMap {TRIM_FLAG =  0, QI_RESET=  1, PITCH_STICK=  2,
                 ROLL_STICK=  3, PEDALS  =  4, PITCH_RATE =  5,
                 ROLL_RATE =  6, YAW_RATE=  7, AOA        =  8,
                 BETA      =  9, TAS     = 10, QBAR       = 11,
                 THETA     = 12, PHI     = 13, ACCEL_X    = 14,
                 ACCEL_Y   = 15, ACCEL_Z = 16};
  enum OutputMap {RUDDER_CMD  = 0, AILERON_CMD= 1,
                  ELEVATOR_CMD= 2, QI_INPUT   = 3}

  ExampleModel(Aircraft* parent_aircraft);
  virtual void update();
  double getRudderCmd() const {getRootOutput(RUDDER_CMD);};
  double getAileronCmd() const {getRootOutput(AILERON_CMD);};
  double getElevatorCmd() const {getRootOutput(ELEVATOR_CMD);};
  void putPitchStick(double x) {putRootInput(PITCH_STICK,x);};
  void putRollStick(double x) {putRootInput(ROLL_STICK,x);};
  void putPedals(double x) {putRootInput(PEDALS,x);};

private:
  Aircraft* aircraft;  // Pointer to parent aircraft.
  double    qi_reset;  // Reset value for pitch law integrator.
}
```

Figure 4 Header for ExampleModel

Client Code

Figure 4 and 5 provides the client code that integrates the model into the LaSRS++ simulation[**]. In this example, the client code is a class derived from the SimulinkModel class. Figure 4 shows the header file. Enumerations are used to name the indices into SimulinkModel's I/O vectors. The enumeration simplifies code changes if later revisions of the model reorder the root-level "port" blocks; only the value of the enumeration must be changed to correct the client code. The index for putRootInput() starts with zero, following C++ convention; it does not start with one which is Simulink convention. The last three elements of the input vector are the three acceleration components represented by the "Accels_g" block. Thus the example illustrates that the indices of the SimulinkModel vectors do not normally match the root-level port numbers in the Simulink model but the order is maintained. The class provides mutator methods for the pilot commands that directly inject the values into SimulinkModel's input vector. Likewise, the class includes accessor methods for the control surface commands that directly extract data from SimulinkModel's output vector.

Figure 5 shows the body file for ExampleModel class. The body file includes C-style external declaration of the model registration function, simple_control_law. The ExampleModel constructor passes the registration function to the constructor for SimulinkModel. It also configures SimulinkModel to obtain its Mode and Timer information from the parent aircraft. It disables external mode and data logging. ExampleModel redefines the update() behavior that it inherits from SimulinkModel. Example-Model::update() sets the TRIM flag appropriately. It also injects the currently computed trim value for the pitch law integrator. It retrieves data from the parent aircraft and copies it to SimulinkModel's input vector. For brevity, the code does not show the complete list of putRootInput() calls. Then, the method calls SimulinkModel::update() to run the RTW code. If in TRIM mode, the method calculates a new trim value for the pitch law integrator; the derivative input to the integrator is the error in the algorithm. The method concludes by checking whether the auto-code reported any errors and terminates the simulation if any were found. The only step now remaining is to add the ExampleModel code and RTW auto-code to the LaSRS++ project makefile and build the simulation.

Conclusions

The resultant LaSRS++/RTW integration is simple to reuse. The basic interface consists of a single class called SimulinkModel. SimulinkModel reduces the

[**] For brevity, the code shown is simplified for the discussion. It does not accurately reflect the code, as it would appear in a LaSRS++ simulation. But it does represent the level of work required for the client code.

```
#include "ExampleModel.hpp"
#include "Aircraft.hpp"

extern "C" { // Registration function prototype.
  extern SimStruct* simple_control_law(void);
}

ExampleModel::ExampleModel(Aircraft* parent_aircraft):
  SimulinkModel(parent_aircraft->getMode(),
                parent_aircraft->getTimer(),
                simple_control_law, false, 0, false, false),
  aircraft(parent_aircraft),
  qi_reset(0.0)
{}

void ExampleModel::update() {
  // For brevity, not all putRootInput calls are shown.
  putRootInput(TRIM_FLAG , getMode() == TRIM);
  putRootInput(QI_RESET  , qi_reset);
  putRootInput(PITCH_RATE, aircraft->getQ());
  putRootInput(AOA       , aircraft->getAlpha());

  SimulinkModel::update();  // Executes the model

  // Trim algorithm for illustration.
  if (getMode() == TRIM) {
    qi_reset += -0.1 * getRootOutput(QI_INPUT);
  } else {
    qi_reset = 0.0;
  }
 if (error()) exit(-1);
}
```

Figure 5 ExampleModel Body File

auto-code to a system with one input vector and one output vector. SimulinkModel has only four behaviors: construction, initialization, update, and destruction. The registration function that is passed as a constructor argument determines the Simulink model that the SimulinkModel object encapsulates. The SimulinkModel class also provides limited support for RTW's data logging and external mode features. The design was realized with minor modification to RTW; in fact, only one TLC modification was made to force reset integrators to use the reset input when the reset switch was set to high. Also, the RTW library source had to be modified to accommodate multiple Simulink models in a simulation. Establishing guidelines for Simulink model construction and auto-code generation solved most of the integration issues. The SimulinkModel class will work with any Simulink model that follows the guidelines. The Simulink user must generate auto-code using the "Generic Real-Time Malloc" target but no special steps were added to the code generation process. The simulation developer writes only a small amount of code to integrate the SimulationModel object. This code sets construction arguments, calls operations at the appropriate time, populates the input vector, extracts data from the output vector, and performs error checking.

Bibliography

[1] Leslie, R.; Geyer, D.; Cunningham, K.; Madden, M.; Kenney, P.; Glaab, P. *LaSRS++: An Object-Oriented Framework for Real-Time Simulation of Aircraft.* AIAA-98-4529, Modeling and Simulation Technology Conference, Boston, MA, August 1998.

[2] Quatrani, Terry. *Visual Modeling With Rational Rose and UML.* Addison-Wesley. Reading, MA, 1998. ISBN 0-201-31016-3.

[3] Stroustrop, Bjarne. *The C++ Programming Language.* Addison-Wesley Publishing Company, Reading, MA, 1997. ISBN 0-201-88954-4.

[4] Neuhaus, J. *An Object-Oriented Design For Trim.* AIAA-2000-4388, Modeling and Simulation Technology Conference, Denver, CO, August 2000.

[5] Hunt, G. *Real-time Constraints for Simulink 2.* Internal memo, NASA LaRC, March 1999.

[6] *Real-Time Workshop® User's Guide*, The Mathworks, Inc., January 1999.

[7] Booch, Grady. *Object-Oriented Analysis and Design With Applications.* The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994. ISBN 0-8053-5340-2.

[8] Dellinger, W.; Salada, M.; Shapiro, H. *Application of Matlab®/Simulink® to Guidance and Control Flight-Code Design.* AAS 99-062, Proceedings of the 22nd Annual AAS Rocky Mountain Guidance and Control Conference, Breckenridge, CO, February 1999.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37266

AIAA-2000-4392

DYNAMIC-WINDOW SEARCH FOR UNEVENLY-SPACED PARAMETER TABLES

Steve Sugjoon Yoon* and Hyunjoo Kang†
Sejong University, Seoul, Korea

Chul Jung Kim‡
Korea Research Institute of Standards and Science, Taejon, Korea

ABSTRACT

Various parameter values are provided in the form of tables, where data keys are ordered and unevenly spaced in general, for real-time simulation of system components or dynamics of vehicles such as airplanes, automobiles, ships, and so on. The main purpose of this study is to compare conventional searching algorithms and to find or develop the most efficient searching method under constraint of real-time simulation, especially hardware-in-the-loop simulation. Since the real-time constraint enforces use of a fixed step size in integration of system differential equations because of the inherent nature of input from and output to real hardware, the worst case of iterated probes in searching algorithms is the key measure of comparison. If a parameter value has certain dynamics because of its relation with the state variables of the simulated system, the integration algorithm and the step size, a searching region at a given time frame can be reduced dramatically from the entire data table taking advantage of the information. The size of the reduced searching region, named Dynamic-searching window, varies and the window moves by its own dynamics as simulation time runs. Numerous numerical experiments were conducted with various data tables of different sizes and types, and yielded results compatible with relevant theories. In the case of non-linear data distribution, Bisection search turns out to be superior to Interpolation search and its derivatives. But, the logic reverses for linear or near linear data distribution, because interpolation searches locate key values at once. Conventional searching algorithms should be combined with the new Dynamic-window search in order to guarantee stable and fast search of parameter values.

1. INTRODUCTION

A mathematical models for the simulation is defined with differential equations, and function generation [1] is processed with given data tables in each integral step in order to compute necessary parameter values. The function generation is composed of search and interpolation procedures. The searching process may become one of key factors to determine over-all performance of simulation, if given data tables are very large and complicated. For example, millions of polygons modeling terrain surface of a region may have to be searched to generate ground reaction force for an off-road vehicle simulation. The main objective of this paper is to develop more effective searching algorithms than conventional ones, which are applicable to real-time

* Associate Professor, sjyoon@sejong.ac.kr
Member AIAA., Dept. of Aerospace Engineering
† Research Assistant
‡ Principal Researcher , Division of Technology
Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

simulation, especially hardware-in-the-loop simulation (HILS). For the purpose, conventional searching algorithms are compared theoretically and experimentally from the viewpoint of searching speeds and the criterion is searching speeds. The searching speeds may be compared in the worst cases and expected values of probe numbers to locate a key value. Among the two measures, the worst case of searching iteration is more critical in HILS applications, since fixed integration step sizes are used to guarantee input from and output to hardware at expected times and the probe numbers are limited by a given step size.

In the field of database, very effective searching algorithms such as Hashing function[2]. Block search[3], Fibonnacci search[3], etc. are used to search for stored data keys, but are not suitable for unit-interval search. In real-time simulation, especially HILS, well-known Bisection search[4], Interpolation search[5], and Fast search[6] have been used without profound understanding of the algorithms. Since the searching principle for discrete data tables is fundamentally the same as that of root computation of algebraic equations, Modified regular falsi method[7], Secant method[7], and Newton's method[7] in numerical analysis can be modified for discrete data search, too. Among them, Secant and Newton's methods are out of concern because searching or root location may fail depending on initial values of search. Thus, only Bisection, Interpolation, and Fast searches are compared in this study.

Dynamic-window search, is proposed in this paper, which can be combined with the conventional algorithms to accelerate their searching speeds. Given parameter values vary according to a certain implicit dynamics related to state variables in most dynamic systems. If not, applied integration algorithms and frame sizes indirectly provide parameter values with a certain dynamics. Dynamic-window search takes advantage of the information of implicit parameter dynamics, and set a searching window limiting candidate data keys at each integration step. The reduced searching region, with which searching is processed at each step, accelerates searching speed. This explains why Dynamic-window search is compatible with conventional searching algorithms.

One series of numerical tests are made in order to exploit the characteristics of conventional searching algorithms such as Bisection, Interpolation, and Fast searches. Three different types of data tables, where data keys are evenly spaced in the form of either 1st- or 2nd - or 3rd- order polynomial, are used for numerical comparison of conventional searching algorithms. The other series of numerical experiments are to evaluate the effect of Dynamic-window search in searching speeds. The data table used for the comparison is a 2-dimensional data table with 1,000 X 1,000 data keys evenly distributed in sinusoidal forms. The numerical tests show consistent results as follows. The speed of Bisection search turns out to be affected not by types of data distribution, but only by the number of stored data keys. If the criterion is actual execution time for searching, Bisection search turns out to be superior to the others in most numerical tests. Interpolation search and Fast search show better performance in the case of linear or near linear data distribution than Bisection search. If the conventional searching algorithms are combined with Dynamic-window search, their searching speeds can be improved dramatically. Dynamic-window search is especially effective with Interpolation search and Fast search, since it not only reduces the sizes of searching regions at each step, but also makes their data distributions linear. In

conclusion, given parameter tables should be carefully examined from the viewpoint of their data distribution in order to determine the most effective searching algorithm for the application, and Dynamic-window search should be combined with the algorithm to accelerate its own searching speed.

2. COMPARISON CRITERIA

Typical simulation of a dynamic system requires a series of searching processes for unit intervals including pursued parameter values, not stored key values in a given data table. There exist several methods to index searching speeds, such as execution time $T(n)$ for searching in an 1-dimensional data table with n keys, upper limit *Big O*[8] and lower limit Ω[8] of $T(n)$. The upper limit *Big O* is of special concern in real-time simulation, neither the expected value nor the lower limit of probe numbers. The *O*-expression (*Big O*) is determined by counting the maximum number of probes in order to locate the unit interval.

It should be noticed that actual execution time of a searching algorithm on a computer could be a more meaningful criterion than the relevant probe number. If the logic of algorithm *A* is simpler enough than algorithm *B*, the actual execution time for searching algorithm *A* can be faster than that of algorithm *B* even if the probe number of algorithm *A* may be bigger than that of algorithm *B*. That is, the upper limit of necessary probes to locate the unit interval may not be the only criterion in real-time simulation, depending on applications. Another fact, which should be carefully considered in selection of searching algorithms, is that the worst case of a searching algorithm may not occur at all during simulation, depending on the data distribution types of given parameter tables.

3. SEARCHING ALGORITHMS

The following is about the logic of conventional searching algorithms and their *Big O*'s. The analysis is made with respect to a 1-dimensional data table with n keys.

Bisection search

Bisection search halves a given table repeatedly until the unit interval comprising a pursued parameter value is located. *Big O* for Bisection search is $O(\log_2 n)$[1].

Interpolation search

Interpolation search is a variation of Regular falsi, which is a numerical method to determine roots of an algebraic equation. It assumes linear distribution of data keys in a given data table, and locates a pursued unit interval. Depending on types of data distribution, its searching speed varies very much. In the worst case, Interpolation search results in successive search of every unit interval. In the worst case, this method results in successive search of every unit interval. Thus, repeated probes are required to locate the pursed unit interval as many times as the number of unit intervals. That is, its *Big O* is $O(n)$[1]. The algorithm is described as follows:

Algorithm
$L = 1, r = n$
For $j = 0, 1, 2,...$ until satisfied, do:
  Set $m = (N\text{-}D(l)) * (r\text{-}l) / (D(r)\text{-}D(l)) + l$
  If $D(m+1) < N$, set $l = m + 1$
  Else if $D(m) > N$, set $r = m$
  Otherwise 'N' exist in the interval $[l, r]$

, where N represents the pursued parameter value and $D(i)$ the ith key value.

Fast search

Fast search, a variation of Interpolation search, repeatedly reduces the size of a searching region by a certain rule defining the next searching region in order to avoid the case of successive search of every unit interval. Thus, Fast search of $O((\log_2 n)^2)$ is faster than Interpolation search of $O(n)$ in the worst cases [1].

If the size of a neglected region at each probe is fixed, the logic of Fast search becomes much simpler, and the required execution time on an actual computer reduces. Thus, practical improvement of searching speed can be achieved depending on applications. Modified fast search is explained in the following algorithm:

Algorithm
$l = 1, r = n$
For j=0, 1, 2,... until satisfied, do:
  Set $m = (N\text{-}D(l)) * (r\text{-}l) / (D(r)\text{-}D(l)) + l$
  If $D(m+l) < N$, set $l = m + 1$
    If also $|m+1, r\text{-}1| < |m, l+1|$, set $r = r + 2$
    Else set $r = r + 1$
  Else if $D(m) > N$, set $r = m$
    If also $|m+1, r\text{-}1| < |m, l+1|$, set $l = l + 2$
    Else set $l = l + 1$
  Otherwise 'N' exist in the interval $[l, r]$

Fig.1 shows an example of the worst case for Fast search, where slops of the lines connecting the end keys do not change in successive probes. But, Modified fast search reduces 3 unit intervals at least at each probe. Thus, *Big O* is computed by the following rule:

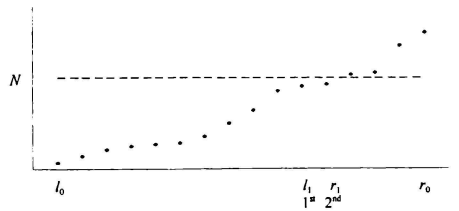
$$n - 3k = 0 \quad (1)$$

$$k = \frac{n}{3} \quad (2)$$

Modified fast search was adapted, rather than the original Fast search, for the numerical tests described in this paper, because the modified one showed a better performance.

4. DYNAMIC-WINDOW SEARCH

The new searching algorithm, Dynamic-window search, can be combined with the conventional algorithms to accelerate their searching speeds. Given parameter values vary according to a certain implicit or explicit dynamics related to state variables in most dynamic systems. If not, applied integration algorithms and frame sizes at least indirectly provide parameter values with a certain dynamics. Parameter values required at each integration frame move around the positions at the previous frame, following the certain dynamics. Dynamic-window search takes advantage of the information of implicit or explicit parameter dynamics, and set a searching window limiting candidate data keys for search at each integration frame. The reduced searching region, within which searching is processed at each frame, accelerates searching speed. The dynamic-searching window moves and its size varies at each integration time frame, while searching is processed



< Figure 1. Fast search.>

within the window. This explains why Dynamic-window search is compatible with conventional searching algorithms. In order to understand how to implement Dynamic-window search and its concept, we assume that a data table of $C_L(\alpha)$ is given and α is a state variable integrated at each integration time frame. The concept is illustrated in Fig. 2. The searching window centered at k_i, whose radius is r_i, is the region to be searched at the *i-th* integration frame in order to determine the wanted parameter value $C_L(\alpha_i)$. Here k_i represents the index of a stored data key and r_i is an integer to define positive difference between the indices of central and boundary data keys in the window. At the *i-th* integration frame, k_i is determined by the rule

$$k_i = \text{index}(\alpha_{i-1}) + \rho_i \cdot \text{int}\left(\frac{\alpha_i - \alpha_{i-1}}{w_{i-1}}\right) \qquad (3)$$

and r_i by the equation

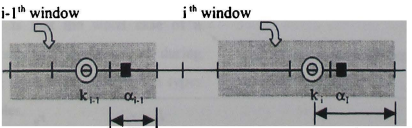
$$r_i = r_{i-1} + \delta_i \cdot \text{int}\left(\frac{\alpha_i - \alpha_{i-1}}{w_{i-1}}\right) \qquad (4)$$

The function, *index*, determines the upper index of the two consecutive keys comprising α in the direction of α's movement, while the function, *int*, yields a closest integer to the argument in the direction of zero. $w_{i-1}$is the size of a unit interval of two consecutive keys, where α_{i-1}resides. The gains, ρ_i and δ_i, are to tune the speed and the size of dynamic searching window. Eqns. (3) and (4) can be viewed as Euler integration or 1st –order Taylor series expansion if ρ_i and δ_i are set to 1 and index (α_{i-1}) replaced by k_{i-1}. Initial values of α and r can be determined off-line before simulation starts. The gains, ρ_i and δ_i, can be set to fixed integers after appropriate information is obtained through enough test runs. Eqns. (3) and (4) can be physically understood that the speed and the size of a dynamic searching window are respectively proportional to the parameter dynamics, which is related directly or indirectly to state variables defining system dynamics. The resultant searching speed depends not on the original table size, but on the reduced window size, $2r_i$. The Dynamic-window search concept can be easily expanded to multi-dimensional parameter tables.

5. NUMERICAL TESTS

One series of numerical tests are made in order to exploit the characteristics of conventional searching algorithms such as Bisection, Interpolation, and Fast searches. The result is summarized in Table 1. The other series of numerical experiments are to evaluate the effect of Dynamic-window search in searching speeds, which is revealed in Table 2.

Three different types of data tables, where 1,000 data keys are evenly spaced respectively in the form of either 1st- or 2nd - or 3rd-order polynomial as shown in Fig. 3, are used for numerical comparison of conventional searching algorithms. The equally spaced data tables are used for convenience, but the same logical results of searching applies to both equally and unequally spaced data tables. The numbers in the row for "Time/probe" of Table 1 represent actual execution time in seconds to perform a probe on a Pentium II PC with 266MHz clock speed



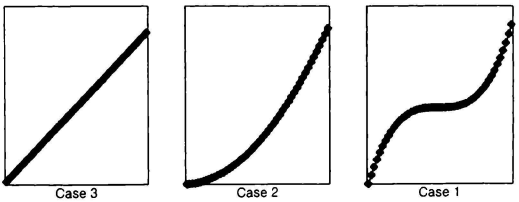
< Figure 2. Dynamic-window search. >

and 64Mb RAM, while the numbers in the upper rows imply required probes to locate unit intervals. That is, the actual computation time required for searching is determined by multiplying the number of probes and computing time per a probe. The row of "Total" represents the averages and the worst iteration numbers of the upper three cases. The numbers were obtained by averaging the results after at least a few scores of test runs. The smaller the numbers are, the faster the searching speeds are. Table 1 shows Bisection search does not depend on types of data distribution, but on the number of stored data keys. Table 1 also shows that Interpolation and Fast searches are affected severely by data distribution types. Their searching becomes slower as relevant data distribution fluctuates more. If the criterion is actual execution time for searching, bisection search is superior to the others in most numerical tests. Interpolation and Fast searches show better performance only in the case of linear or near linear data distribution than the Bisection search.

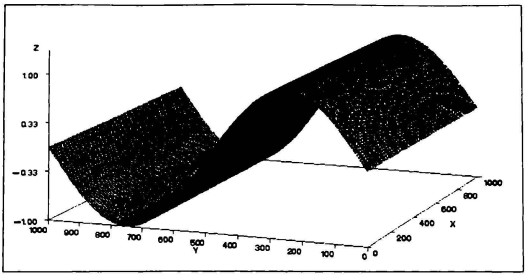
The next numerical tests are to evaluate the effect of Dynamic-window search in searching speeds. Bisection and Interpolation searches are combined with Dynamic-window search respectively, and compared. Probe numbers, to locate 200 parameter values from a 2-dimensional data table with 1,000 X 1,000 data keys evenly distributed in a sinusoidal form [Fig. 4], are listed in Table 2. The gains, ρ_I and δ_I, in Eqns (3) and (4) are set to 1 and the value of r_I set to 0 for convenience throughout the numerical tests. Dynamic-window search noticeably accelerates searching speeds in both Bisection search and Interpolation search. The accelerating effect is even greater for the Interpolation search, because the searching window makes data distribution look linear by narrowing a candidate searching region at each probe. As shown in the aforementioned tests, Interpolation search locates a parameter value at once if the data distribution is linear.

6. CONCLUSIONS

The speed of Bisection search turns out to be affected not by types of data distribution, but only by the number of stored data keys. If the criterion is actual execution time for searching, Bisection search turns out to be superior to the others in most numerical tests. Interpolation search and Fast search show better performance in the case of linear or near linear data distribution than Bisection search. If the conventional searching algorithms are combined with Dynamic-window search, their searching speeds can be improved dramatically. Dynamic-window search is



< Figure 3. Three types of data tables used for comparison of conventional algorithms. >



< Figure 4. Two-dimensional data table used to evaluate the effects of Dynamic-window search. >

Table 1. Numerical comparison of searching algorithms in searching speeds.

	Bisection		Interpolation		Modified fast	
	Mean	Worst	Mean	Worst	Mean	Worst
Case 1 (# of iterations)	9.0	10	1	1	1	1
Case 2 (# of iterations)	9.0	10	16.8	65	16.3	64
Case 3 (# of iterations)	9.0	10	21.8	98	21.5	97
Total (# of iterations)	9.0	10	13.2	98	12.9	97
Time/probe (sec)	0.00031		0.00102		0.00109	

Table 2. Effects of Dynamic-window search in searching speeds.

	Bisection		Interpolation	
	Mean	Max	Mean	Max
without dynamic-window (# of iterations)	8.95	10	6.95	13
with dynamic-window (# of iterations)	3.16	6	1.21	2

especially effective with Interpolation search and Fast search, since it not only reduces the sizes of searching regions at each step, but also makes their data distributions linear. In conclusion, conventional searching algorithms should be combined with the new Dynamic-window search in order to guarantee stable and fast search of parameter.

REFERENCES

1) Yoon S., "A Study of Searching Algorithms for Real-time Simulation of a Dynamic System", Proceedings of AIAA Flight Simulation Conference, Aug. 1996.

2) Warren Burton F. and Lewis Gilbert N., "A Robust Variation of Interpolation Search", Information Processing Letter, Vol. 10(4,5), 5 July

1980, pp.198-201.

3) Lewis Gilbert N., Boynton Nancy J. and Burton F. Warren, "Expected Complexity of Fast Search with Uniformly Distributed Data", Information Processing Letter, Vol. 10(1), 27 October 1981, pp.4-7.

4) Conte Samuel D. and Carl de Boor, "The solution of nonlinear equations", Elementary Numerical Analysis an Algorithmic Approach (3th), McGraw-Hill, 1965.

5) Gonnet G., George J. and Rogers L., "An Algorithmic and Complexity Analysis of Interpolation Search", Acta Informatica, Vol.13, Springer-Veriag 1980, pp.39-52.

6) Leon S. Levy, "An Improved List-Searching Algorithm", Information processing letters, Vol. 15(1), 19 August 1982, pp.43-45.

7) Yehoshua Perl and Reingold Edward M., "Understanding The Complexity of Interpolation Search", Information processing letters, Vol. 6(6), December 1977, pp.219-222.

8) Sedgewick, Robert, "The searching algotithms", Algotithm In C, Addison-Wesley Pub.Co., 1990.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37267

AIAA-2000-4393

IMPLEMENTING DYNAMIC SYSTEM MODELS IN THE ASSET SIMULATION FRAMEWORK

Stephen D. Derry*, Jeffrey Maddalon*

NASA Langley Research Center
24 W. Taylor Road
Hampton, VA 23681

Abstract

The Aircraft System Simulation Environment and Toolkit (ASSET) project is being undertaken as a proof of concept, aimed at demonstrating a rapid model prototyping and modification capability by utilizing modern software technologies and design concepts. Initially, the project is focused on a self-contained batch simulation, but it is designed to ultimately run embedded within other software environments, including real-time. This report focuses on the design and implementation of the model software, which strictly follows a block-diagram design paradigm. The initial implementation, a six-degree-of-freedom simulation of an F-16 airplane, is presented, along with current status and near-term plans.

Introduction

NASA Langley Research Center utilizes flight simulation, both batch and real-time (pilot in the loop), to support research in a number of disciplines, ranging from control system development and handling qualities evaluation to human factors. Because these simulations are used for research rather than operational purposes (such as training), they are subject to frequent change.

Several of the research disciplines require frequent modifications to the simulation model itself, to either the aircraft plant, the associated control system, or both. To support a rapid design iteration rate, the researcher should be able to modify the model and test the simulation in a desktop workstation environment without the support of software specialists. This desktop batch simulation must be capable of producing trim solutions and time-history evolutions; additionally, linearized models, possibly of reduced order, must be extracted from the full non-linear model. Finally, the non-linear model must be portable to a real-time environment for piloted simulation and/or flight test, preferably without recoding.

ASSET will attempt to demonstrate rapid model software development and modification in a desktop workstation batch simulation environment that supports the three primary operations described above (trim, linearization, and time-history evolution).

ASSET Software Overview

To facilitate development of simulation model software by non-software experts, ASSET is designed to be functionally de-coupled and easy to learn. In this context, functional de-coupling refers to the design goal of isolating the software context in which models operate. Model software can then be developed and/or modified without knowledge of the context in which the model is to be run (i.e. whether the model is a stand-alone system or part of a larger system), and with little knowledge of the software architecture of the other parts of the ASSET framework. Functional de-coupling should also enhance the re-usability of the software.

---

* *Computer Engineer, Systems Development Branch, Airborne Systems Competency.*

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

Object-oriented design techniques are employed throughout the ASSET project. To maintain simplicity of design and ease of learning, the class hierarchy (levels of inheritance) is intentionally kept shallow – typically two or three levels deep at the most.

ASSET uses the Java programming language [1]. Java was chosen primarily for its portability, support for the object-oriented design paradigm, and ease of learning, as well as the breadth of industry support and the availability of already-developed software (such as vector and matrix classes and graphical user interface foundational classes).

ASSET consists primarily of six major components: applications, models, a data store, data collectors, a user interface, and the framework of common software. A given simulation project will typically have a unique application and models, but use standard ASSET classes for the remaining components. This paper focuses on the ASSET concept of a model and its software realization. A companion paper [2] discusses the data store and the data collectors.

Initially, ASSET is targeted to run in a stand-alone batch environment, providing a desktop (non real-time) simulation environment with a graphical user interface (GUI). Long-term plans are to embed ASSET into other simulation software environments, including commercial products in a desktop batch environment and, ultimately, real-time environments for piloted simulation. Once ASSET is embedded into a simulation environment, then models developed for ASSET will run in that environment.

When embedding ASSET models into a non-ASSET environment, the embedded ASSET components will typically include the models, data store, data collectors, and portions of the framework, along with interfacing software specific to the environment in which the ASSET components are embedded.

The design of ASSET allows for it to be embedded into a real-time environment, although ASSET itself does not provide a real-time environment. To this end, ASSET does not create any objects in the run-time portion of the software, which eliminates the necessity of running garbage collection in real-time.

Garbage collection is a mechanism used by Java to reclaim unused memory to make it available for the creation of new objects. Garbage collection is undesirable for real-time operations because it is time consuming and its invocation generally occurs at unpredictable times.

One final aspect of ASSET is that it is designed not just for flight simulation. While many components of the framework are specific to flight simulation (equations of motion, etc.), ASSET can be used to support the modeling and simulation of any set of dynamic systems.

Software Implementation of Models

Basic description and class design

The Model super-class is the basic abstraction for all mathematical models in ASSET. A Model object represents a block in a block diagram. Models have inputs, outputs, and perhaps states. Note that a block diagram typically depicts the inputs and outputs for the individual blocks, but the states are not explicitly shown. Similarly, the input and output vectors for ASSET Models are visible to the block's "owner", but the states are generally not. The "owner" of a model object is usually the object which created it.

All classes which implement models are sub-classes (descendants) of the Model super-class, either directly or indirectly. The Model super-class manages certain bookkeeping functions, such as mode and time, whereas the sub-class implements the mathematical state and output equations that govern the behavior of the model.

All models have essentially the same software interface. Most of the methods defined for a model are implemented in the super-class; the remaining methods defined in the sub-class usually over-ride methods of the super-class. Rarely are additional, model-specific, methods needed. The uniformity of the software interface for all models simplifies the software design, both for the models themselves and the upper layers of software that use the models.

Signals and model interface vectors

Model input and output vectors are implemented as Java arrays of `double`'s, with one `double` for each input and output signal. Each of the signals has a

unique name, and both the names and values for the input and output signals are accessible to the model's owner. To improve run-time execution performance, signal name lookups are typically done only once, at construction time, and the corresponding indices are retained for use during run-time.

A model may also contain states, which are defined and maintained by the model and not generally visible outside. ASSET supports both continuous and discrete states, and a given model may contain either or both types. The state vectors are also implemented as arrays of `double`'s with unique names for each state. State and state derivative vectors are maintained for the continuous states, whereas state and next-state vectors are maintained for discrete states. Special methods in the Model super-class provide access to the state-related vectors, primarily to support linear model extraction; these methods are not intended for general use.

An additional interface, the parameter vector, is used by the application to pass signals to the model without intervention by the model's owner. These parameters can be used to adjust gains, etc. for control systems, or to trigger failures or other anomalous events.

Finally, a model may select certain of its quantities to be sampled for recording and analysis purposes; these may include internal signals that are not otherwise visible. The sampling and recording are accomplished via the data store and data collectors and are described in [2]. Other than defining which of its quantities are to be sampled, the model sub-classes do not participate in the sampling process.

Units

All quantities in ASSET are internally maintained (i.e. in variables) in a set of coherent internal units. A set of units is coherent if it has exactly one unit for each type of quantity, and is mutually related by multiplication and division without conversion factors [3]. For example, the area unit would be the distance unit squared, and the unit of velocity would be the distance unit divided by the unit of time.

The modeler should make no assumptions about what actual units are used internally–as long as all quantities are maintained in internal units, the results will be consistent. Whenever a numerical quantity is used, whether as an input, output, or a hard-coded constant, the corresponding unit conversion must be performed so that the quantity is stored internally with the correct coherent unit.

ASSET can be configured to use either of two internal coherent unit systems: the International System of Units (SI) [3] or a customary set of units based on the foot, slug, second, and degree Rankine. As long as all unit conversions are properly specified, the actual selection of the internal unit system is transparent and arbitrary. The SI system is generally used.

Embedding models

Just as a block in a block diagram may itself be specified as a lower-level block diagram, an ASSET model may embed models within it. In this case, the embedded model is a separate object which is "part of" the containing model. The containing model uses the same interface (input and output vectors, etc.) and methods to communicate with the embedded model.

The software that implements the embedded model has no dependency at all on the containing model. This means that models may be designed with no knowledge of the context(s) in which they are to be used, whether they are to stand alone or to be part of a larger system. This aspect of ASSET's design facilitates testing as the model software is not changed between unit testing and incorporation into a larger system. It also facilitates reuse since the model software is not sensitive to its context.

A model's states are often contained in its embedded models. The ASSET framework will include basic models, such as integrators and transfer functions, which are typically employed to manage the states of a larger model. Due to the uniform model interface, it is transparent to a model's owner whether the model's states are contained directly in the model or within embedded models. To maintain this transparency, the special methods (required to support linearization) for accessing and manipulating a model's state vector are recursive: they operate on all states contained in the model itself and in all embedded models.

Each model is given a name by it's owner; this name is unique among its siblings (i.e. among all models created by the same owner object). Each model also has a hierarchical name which is unique among all models defined in the system. The hierarchical name for a model is its given name appended to the hierarchical name of its parent (containing) model, if any. For example, the name of an aileron model contained within a standalone model of an open-loop F16 aircraft might be "`F16OpenLoop.Aileron`".

Model execution cycle

A model execution cycle starts with the model's owner establishing the appropriate quantities in the model's input vector. The owner then invokes the `runModel()` method which is defined in the Model super-class. The `runModel()` method performs the following actions, if required, depending on the current mode (modes are described in a separate section below) and other model settings:

1. Advance time
2. Propagate states to the new time by invoking the sub-class method `propagate()`
3. Invoke the sub-class method `calculateModel()`
4. Invoke model sampling via the DataStore

After this completes, the model's current outputs are available in its output vector.

The `calculateModel()` method, which is implemented in the sub-class for each specific type of model, begins by initializing the model's states if required for the current model mode. The initial values for the states are determined either by forcing steady-state conditions based on the current inputs, or by utilizing special input signals which serve solely to provide state initialization data from outside the model (see Model residualization below). After the states are initialized (if necessary), the primary model calculations are performed: state derivatives for continuous states, next values for discrete states, and model outputs. These calculations should be the same, regardless of the model mode. This helps insure that the model behavior is consistent between modes by minimizing special "reset" code. If the model embeds other models, the `runModel()` method should be invoked on the embedded models.

The `propagate()` method, which is also implemented in the sub-class for each type of model, propagates the states to the next time step. Continuous states are integrated using the state derivative calculated for the previous step (and perhaps other data, depending on the specific integration method used), and the discrete states are advanced to their next values which were also calculated in the previous step. The `propagate()` method need only be defined if the model directly maintains its own states. Frequently, states are embedded in standard models such as filters and integrators, in which case an implementation for `propagate()` is not necessary.

The ASSET framework currently includes a small set of standard integrators and transfer functions which are implemented using single-pass algorithms suitable for use in a real-time environment. This set of integrators and transfer functions will be expanded as necessary, and it is anticipated that these standard models will be used to maintain most states.

Time management

A variety of time and mode management services are provided by the Model super-class. The primary time management functions are the maintenance of the time and time step for each model. The time step for top-level models (those which are not embedded inside a containing model) is established by the model's owner. By default, embedded models inherit the time step of their containing model; however, the time step may be set to a multiple or a sub-multiple of the containing model's in order to sub-rate or super-rate the model. This capability allows different parts of the system to run at different rates, transparently to the model and application software.

Super-rating an embedded model means that several cycles of the embedded model are run for each cycle of the containing model. This is useful for modeling a subsystem which contains higher frequency dynamics than the full system without requiring the entire system to be modeled at the smaller time step. Sub-rating is the opposite: the embedded model is only evaluated once for several cycles of the containing model. Sub-rating is primarily provided to reduce execution time for subsystems whose dynamics are appreciably slower than

the system as a whole. To make sub-rating useful for a real-time environment, a set of sub-rated models may be phased. For example, a model of a twin-engine airplane may run its engine models on alternating time steps. Super-rating and sub-rating are handled entirely by the Model super-class.

Modes

ASSET models can be in one of three modes: Initialize, Steady, and Dynamic. When a model is in Initialize mode, the states are initialized prior to the calculation of the state and output equations, and time does not advance. In Steady mode, the states are initialized similar to the Initialize mode, but time advances. In Dynamic mode, time and states are advanced and integrated prior to the calculations. Other than the state initializations for the Initialize and Steady modes, the mode should be transparent to the modeler and the model sub-class. In addition to simplifying the software, this helps insure consistent results.

The Initialize mode is used to establish and/or trim the initial conditions, while the Steady and Dynamic modes are used to produce a time-history evolution starting from the initial conditions. A model in the Dynamic mode will respond according to the dynamics described by its state and output equations, while a model in the Steady mode is forced to produce a steady-state response as described in the next section (model residualization).

To initialize the system, all models are placed in the Initialize mode. To run the system (over time), all models are placed in either the Dynamic or Steady mode. Normally, all models will be set to Dynamic mode; however, selected models may be placed in Steady mode to reduce the complexity of the dynamics being studied. For example, actuator or engine models could be placed in Steady mode so that their steady-state response is produced instantaneously while removing their transient response. This can be used to produce a reduced-order linear model of the aircraft.

While running (i.e. not initializing), the system as a whole may also be suspended, which means that the passage of time and propagation of states does not occur. While suspended, only the normal state and output calculations, including (re)initialization of states if in Steady mode, are performed by the `runModel()` method. This feature is used to produce linear models of the system by allowing the application to perturb each state and input, in turn, and observe the system response in the output vector and the state derivatives. Special methods are provided by the Model super-class to access the state and state derivative vectors for this purpose. Note that although only the top-level model's inputs and outputs are visible, the state and derivative vectors include the states and derivatives of all embedded models.

Suspension and the special state vector access methods could also be used to support the use of an external integration algorithm, although this has not been fully designed yet. (Additional methods would be needed in order to manually adjust the time and time step.) Such external integration methods would be implemented separately from the model and would allow the user to provide alternate integration algorithms, including multi-pass algorithms such as Runge-Kutta, possibly with variable step size.

Model residualization

In ASSET, certain models are considered to be residualizable, meaning that the states can be initialized such that the model is placed in a steady-state condition for the current inputs. Steady state condition is defined to occur when the residual error is zero, where the residual error is the root-sum-square of the continuous state derivatives and the discrete state differences (next state – current state).

This feature allows the user to optionally remove the dynamics of selected residualizable subsystem models, such as aircraft actuators, by placing the subsystem models in Steady mode while the rest of the system is in Dynamic mode.

One area of difficulty for determining a steady-state solution is that some models contain implicit loops in their state equations.[1] A standard approach is to iterate

[1] Implicit state equations occur when the state derivatives are computed as functions of several quantities, including the state derivatives or equivalent forms. This

the calculation of the states and derivatives until the residual error is driven suitably close to zero.

Classes which implement residualizable models identify themselves to the Model super-class as such. This enables the super-class to automatically calculate the residual error and iterate the invocation of `calculateModel()` if the model is in Initialize or Steady mode.

Note that models which are not residualizable require special inputs for initializing the states. Since these states require external participation in their initialization, iterative calls to `calculateModel()` are not performed on these models.

Signal wiring

One of the more time-consuming parts of writing software for models in a block-diagram paradigm is the passing of signals from one model to another, i.e. wiring the connections. For example, ASSET's model for the aircraft rigid equations of motion has over two hundred input and output signals.

ASSET provides several framework classes to assist with this chore. One set of classes implement busses, which aggregate a number of signals. Busses provide a structure for managing the names for a group of signals, and for mapping the names so that the two sides of the "connection" can name the signals differently. Input, output, state, and parameter vectors can be defined in terms of busses as well as individual signals. Standard busses are defined for basic aggregate quantities, such as vectors, matrices, and quaternions, which are used frequently and are composed of multiple signals. Another class, the Signal List, automates the transfer of quantities between the interface vectors and primitive data types which are used in the model code.

If a model can be described completely in terms of a set of embedded models wired together, the CompositeModel super-class can be used to gain significant leverage in the coding of such model classes. CompositeModel, which is a sub-class of Model, manages the execution of such models. In this case, the model developer merely has to define a sub-class of CompositeModel with a constructor; no other methods need to be defined in the sub-class. The constructor must create the embedded models, establish their wiring and execution order, and describe the data which should be included in the sampling. The CompositeModel superclass provides the methods, such as `calculateModel()`, which are needed for run-time execution. Standard algebraic models, such as gains, summers, and limiters, have been added to the framework to support the use of composite models.

External model interfaces

The block-diagram paradigm, with successive levels of decomposition of a model into embedded models which are "part of" the containing model, has its limitations and cannot be used as a basis for all inter-model communications. Namely, there is occasionally a need for models which are not part of the same system to communicate (i.e. exchange data) with each other. For example, consider the case of an aircraft and an atmosphere. Models can be built for each, but neither entity is "part of" the other; each can exist without the other, but the presence and characteristics of one has a great influence on the behavior of the other!

ASSET uses the Java interface mechanism[2] so that one model can invoke specialized methods of another model. This use of the Java interface adds flexibility to the object-oriented design of the modeling software. However, because it is a special-purpose mechanism, its use is kept to a minimum to avoid excessive complexity in the software design.

Early Implementation – F16 Simulation

The first substantial system to be simulated in ASSET is a simplified F-16 model, described in [4]. This system provides an illustrative example of a hierarchy of embedded ASSET models with several external interfaces.

is common in aerodynamic models, where aerodynamic coefficients are functions of the angle-of-attack derivative and pitch-rate derivative, for example.

[2] Java interfaces basically provide declarations of methods, including parameters and return types, without defining the implementations of those methods. A Java class can be declared to implement an interface, which means that it must define an implementation for all of the methods declared in the interface.

An object diagram for this system is shown in Figure 1, and a class diagram is shown in Figure 2. Note that all of the model classes are either sub-classes of Model or CompositeModel. All of the states in this system are maintained in the KinematicBody and Integrator objects.

The six-degree-of-freedom rigid-body equations of motion are implemented by the AircraftRigidEOM class, which is a sub-class of CompositeModel. These equations are mathematically similar to those used in the current piloted flight simulation framework at Langley, LaSRS++. LaSRS++ is described in [5] and is written in C++ using object-oriented techniques.

The DynamicRigidBody model receives as inputs the external forces and moments acting on the aircraft as well as the aircraft mass and inertia, and computes the resulting inertial accelerations, both translational and rotational. These signals are then passed to a KinematicBody model, which maintains and propagates the aircraft's position and velocity states, both translational and rotational. All of the kinematic states are referenced to an earth-centered inertial Cartesian coordinate system. KinematicBody has built-in integrators for maintaining the rotational and translational velocities (second-order Adams Bashforth) and translational position (truncated Taylor series). It maintains the aircraft attitude as a quaternion, which is propagated by a discrete algorithm described in [6].

Both the DynamicRigidBody and KinematicBody classes interface with an external World model for calculating world-relative positions, velocities, and accelerations as well as the gravity vector. This allows a variety of world models to be used without modifying the basic dynamics classes. Currently, only a flat earth model is implemented, but an ellipsoidal earth model will be developed in the near future.

AirFlow is a CompositeModel which embeds an AirPath model and a GasDynamics model. The AirPath model interfaces with an external Wind model to compute the aircraft's air-relative velocity, given its current world-relative velocity. The GasDynamics model computes such quantities as Mach number, equivalent and calibrated airspeeds, dynamic and total pressures, and total temperature. It takes the true airspeed as an input, and interfaces with an external Atmosphere model to obtain the ambient atmospheric characteristics (static temperature and pressure, density and density gradient, and speed of sound) at the current location of the aircraft. The AirPathAccel model is used to calculate the derivatives over time of the air-relative velocity, such as the derivatives of the angles of attack and sideslip. It is a separate model because the aircraft accelerations are not available when the AirFlow model is invoked.

The current atmosphere model is based on the 1976 U.S. Standard Atmosphere, with provisions for local temperature and pressure deviations. The only wind model implemented at this point is one which provides a constant wind. Additional wind models, along with stochastic turbulence models, are anticipated to be added in the future.

The simplified F-16 vehicle model is taken from [4], and consists of several ASSET models, all embedded within the F16OpenLoop class which is a CompositeModel.

The aerodynamic forces and moments are computed in the F16Aerodynamics class. The aerodynamics model is non-linear and includes eighteen one- and two-dimensional function table lookups with linear interpolation.

The engine dynamics and thrust are computed in the F16Engine class. The engine model includes three two-dimensional thrust tables, and the engine dynamic state is maintained by an ImplicitTrapezoidalIntegrator which is embedded in the F16Engine model.

The three aerodynamic control surfaces defined in the aerodynamics model are positioned by three instances of the ActuatorOrder1 model (namely the aileron, elevator, and rudder objects), each of which embeds an ImplicitTrapezoidalIntegrator.

Any combination of the engine and/or actuator models may be residualized in order to eliminate the effect of their dynamics and thereby reduce the complexity of the system as a whole. This feature may be used to reduce the model for linearization and/or time-history evolution.

Finally, the aircraft mass and inertia properties, as well as the location of the CG, are produced by the F16MassGeometry model (the location of the CG is set via a parameter to this model). The F16SumForces model combines the forces and moments from the engine and aerodynamics models and transforms the moment to the actual aircraft CG, and AircraftRigidEOM is invoked to compute the dynamics of the aircraft.

This implementation consists of a system of twenty-two ASSET models, including a hierarchy of nineteen models for the F-16 aircraft and three independent models for the environment.

Current Status

The portions of the ASSET framework that are essential for simulating 6-DOF rigid aircraft in a simplified environment (flat earth with constant wind) have been developed, along with two simple aircraft models (the F-16 mentioned above and a linearized model of a North American Navion). The Navion model and the rigid aircraft six-degree-of-freedom equations of motion have successfully passed an initial validation by comparison against an established simulation described in [7].

To date, ASSET development activities have taken place on Windows NT 4.0 platforms using the Sun JDK 1.2.2 Java development environment, which includes a just-in-time compiler. Performance measurements of the F-16 model running on a 550 MHz. Intel Pentium III with 256 MB of RAM have resulted in the following times:

230 microseconds per step with sampling each step
195 microseconds per step without sampling

The entire F-16 model, including equations of motion, was exercised each step, and the sampling recorded 210 signals. These tests were performed in a batch environment with no user interface.

Near-term development plans include a graphical user interface, implementation of simple control systems to demonstrate closed-loop simulation, and incorporation of trim and linearization capabilities. (Linearization has already been demonstrated in ASSET on a simple linear actuator model.) Following this, one or more additional aircraft models which are of current research interest will be developed in ASSET in order to assess the software development effort required and to demonstrate ASSET's usefulness in supporting Langley's research mission.

Conclusions

Progress with ASSET to date demonstrates that the block diagram paradigm is a useful design for simulation model software, and that Java is a viable platform for simulation software development. Follow-on efforts will seek to determine ASSET's effectiveness in reducing model software development time.

References

[1] Arnold, Ken; Gosling, James. *The Java Programming Language, Second Edition.* Addison Wesley Publishing Company, Reading, MA, 1998. ISBN 0-201-31006-6.

[2] Maddalon, Jeffrey; Derry, Stephen. *Data Management in the ASSET Simulation Framework.* AIAA 2000-4500. Modeling and Simulation Technology Conference, Denver, CO, August 2000.

[3] Taylor, Barry N. *The International System of Units (SI).* NIST Special Publication 330, National Institute of Standards and Technology, August 1991.

[4] Stevens, Brian L.; Lewis, Frank L. *Aircraft Control and Simulation.* John Wiley and Sons, Inc., New York, NY, 1992. ISBN 0-471-61397-5.

[5] Leslie, R.; Geyer, D.; Cunningham, K.; Madden, M.; Kenney, P.; Glaab, P. *LaSRS++: An Object-Oriented Framework for Real-Time Simulation of Aircraft.* AIAA-98-4529, Modeling and Simulation Technology Conference, Boston, MA, August 1998.

[6] Barker, Lawrence E. Jr.; Bowles, Roland L.; Williams, Louise H. *Development and Application of a Local Linearization Algorithm for the Integration of Quaternion Rate Equations in Real-Time Flight Simulation Problems.* NASA TN D-7347. December 1973.

[7] Jackson, E. Bruce; *Manual for a Workstation-Based Generic Flight Simulation Program (LaRCsim) Version 1.4.* NASA TM-110164, May 1995.

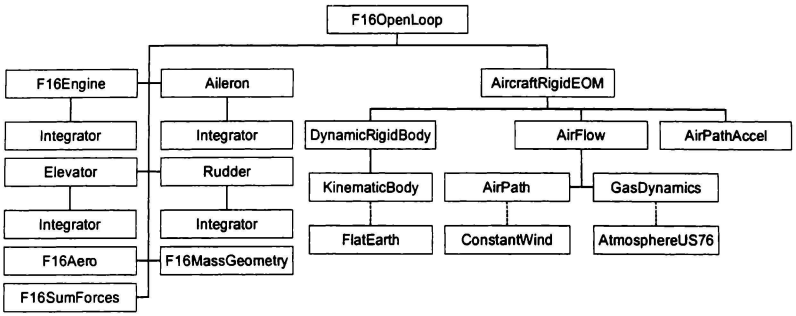


Figure 1. F-16 Model Object Diagram

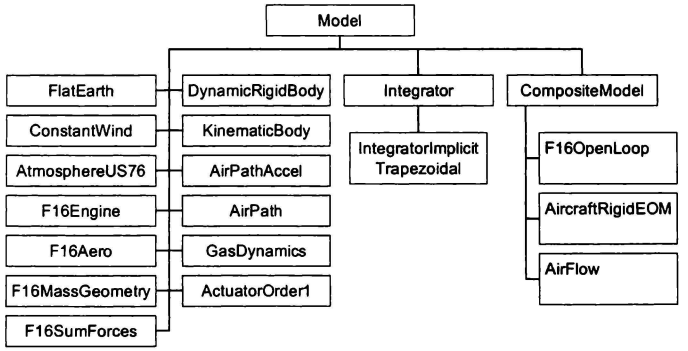


Figure 2. Model Class Diagram

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37268

AIAA-2000-4394

AN EXPERIMENTAL METHOD FOR ESTIMATING DYNAMIC ERRORS IN REAL-TIME SIMULATION

R.M. Howe, Associate Fellow
Applied Dynamics International, Ann Arbor, Michigan

Abstract

This paper describes a methodology for estimating the numerical errors in real-time simulation based on comparing two real-time simulations, one with the nominal integration step size and the second with an augmented step size. Accurate interpolation formulas are then used to convert the outputs from the second simulation to outputs at the discrete times for the first simulation. This permits a direct computation of the difference between the two simulations at common discrete times. Assuming that the dynamic errors vary as h^k, where h is the integration step size and k is known, a simple formula can then be used to calculate the deviation of either simulation from the solution for zero step size. While only approximate, this procedure results in dynamic error estimates that are surprisingly accurate, as demonstrated in the paper in example simulations that use both single and multi-rate integrations. It turns out that three simulations, each with a different integration step size, can be used to estimate the exponent k in the dependence of dynamic errors on h^k. This in turn allows programming errors, such as the inadvertent introduction of a one-frame delay within the simulation code, to be identified. The author feels that the methodology described in the paper can play a major role in optimizing as well as verifying the dynamic fidelity of real-time, hardware-in-the-loop simulations.

1. Introduction

The determination of numerical errors in the real-time simulation of dynamic systems is often ignored because of the difficulty in calculating the errors in a real-time environment. Not infrequently the real-time simulation outputs are merely examined for "reasonableness" in order to confirm the validity of the simulation, even though this approach is highly qualitative. One method that has been utilized is the comparison of real-time simulation outputs with the results from a non real-time reference solution using a much smaller integration step size, although this approach is in general not compatible with real-time simulation inputs. The author has proposed on-line calculation of the approximate local integration truncation error as a tool for measuring real-time dynamic errors[1,2]. However, the relationship between local truncation errors and global errors is at best tenuous, and is highly dependent on the system being simulated.

When numerical integration is used in simulating dynamic systems, the global dynamic error for sufficiently small integration step size h will be proportional to h^k, where k is the order of the numerical integration algorithm. For a given simulation run with two different step sizes, h_1 and h_2, with associated data-point outputs X_{1_r} and X_{2_r} at the discrete time t_r, we can then write

$$X_{1_r} - (X_r)_{\text{exact}} \cong C_r h_1^k \ , \quad h = h_1 \ , \tag{1}$$

$$X_{2_r} - (X_r)_{\text{exact}} \cong C_r h_2^k \ , \quad h = h_2 \ . \tag{2}$$

By subtracting Eq. (1) from Eq. (2), we can solve for the proportionality constant C_r in terms of $X_{2_r} - X_{1_r}$. Substituting the resulting formula back into Eq. (1), we obtain the following formula for the estimated dynamic error at time t_r:

$$X_{1_r} - (X_r)_{\text{exact}} \cong \frac{h_1^k}{h_2^k - h_1^k}(X_{2_r} - X_{1_r}) \tag{3}$$

Eq. (3) shows that the dynamic error, $X_{1_r} - (X_{1_r})_{\text{exact}}$, can be calculated directly from the two solutions, X_{1_r} for $h = h_1$ and X_{2_r} for $h = h_2$. Normally we let the time step associated with the discrete time t_r be equal to the integration step size h_1 used for the first solution, so that for the nth integration frame, $X_{1_n} = X_{1_r}$. The output data points X_{2_n} for the second solution with step size h_2 can then be converted to data points X_{1_r} using interpolation of order at least k. We now consider a simple example.

2. 2nd-order System Example

To illustrate the above methodology for calculating the dynamic errors in a simulation, we initially utilize a 2nd-order linear system with state equations given by

$$X' = Xdot \ , \quad Xdot' = R(t) - X - 2\zeta Xdot \ . \tag{4}$$

Here X is the system output and $R(t)$ is the system input. When the AB-2 second-order formula, a popular integration method for real-time simulation, the difference equations for the simulation are the following:

Copyright © 2000 The American Institute of Aeronautics and Astronautics Inc. All rights reserved.

$$X_{n+1} = X_n + h(1.5Xdot_n - .5Xdot_{n-1}) \ ,$$
$$Xdot_{n+1} = Xdot_n + h(1.5F_n - .5F_{n-1}) \ , \quad (5)$$
$$F_n = R_n - X_n - 2\zeta Xdot_n \ .$$

For studying the simulation accuracy, we will let the input $R(t)$ be the $1-\cos(t)$ function shown in Figure 1. For this input the figure shows the exact response of the continuous system when the damping ratio $\zeta = 0.4$. Note that this input exhibits no initial discontinuity in displacement or slope, either of which can lead to dominant dynamic errors when using predictor integrator methods such as AB-2.

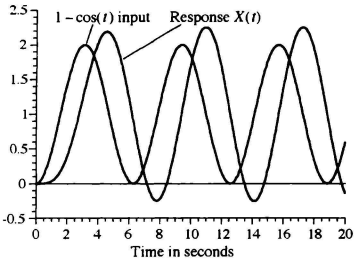


Figure 1. Response of 2nd-order system, $\zeta = 0.4$, to $1-\cos(t)$ input.

We now simulate the 2nd-order system using AB-2 integration with two different integration step sizes, h_1 and h_2. To convert output data points X_{2_n} at the time t_n from the second simulation to output data points X_{2_r} at the time t_r for the first simulation, we employ the following cubic extrapolation formula:

$$X_{2_r} = X_n + (X_{n+1} - X_{n-1})a + (X_{n+1} - 2X_n + X_{n-1})\frac{a^2}{2} + (X_{n+1} - 3X_n + 3X_{n-1} - X_{n-2})\frac{a^3 - a}{6} \ , \quad (6)$$

where

$$a = \text{dimensionless extrapolation interval} = \frac{t_r - t_n}{h_2} \ . \quad (7)$$

In this conversion process, which is performed off-line after the second simulation, we choose t_n such that the extrapolation interval falls in the range $-1 \le a \le 0$. This minimizes the extrapolation error, which is proportional to $X_n'''' h^4$ when using cubic extrapolation.

Since the global simulation errors for AB-2 integration are proportional to h^2, we let the exponent $k = 2$ in Eq. (3). For an AB-2 integration step size equal to 0.1, Figure 2 shows that the errors calculated using Eq. (3) with $h_1 = 0.1$ and $h_2 = 0.11$ are almost identical with the exact errors based on an analytic solution. Similar

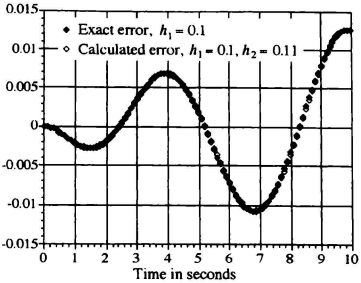


Figure 2. Output error, 2nd-order system. Comparison of exact error with error based on two solutions.

close agreement is observed when the step sizes are doubled, i.e., for $h_1 = 0.2$ and $h_2 = 0.22$. Also, the agreement is good when the frequency ω of the input $R(t) = 1 - \cos(\omega t)$ is equal to values other than unity. We conclude that the experimental method described here for calculating dynamic errors based on running two simulations with slightly different step sizes works quite well when simulating a second-order linear system.

3. 4th-order Dual-speed System

We next consider the 4th-order dual-speed system shown in Figure 3. The system consists of fast and slow subsystems connected in the feedback loop shown in the figure. The fast subsystem has an undamped natural frequency given by $\omega_{n_c} = 50$ and a damping ratio given by $\zeta_c = 0.5$. The overall system can be thought of as a feedback control system with the slow subsystem representing the plant and the fast subsystem the controller, where α is the feedback gain constant and $2\beta\zeta$ is the feedback rate constant. If we neglect the dynamics of the fast subsystem, i.e., let $Y = E$, then the

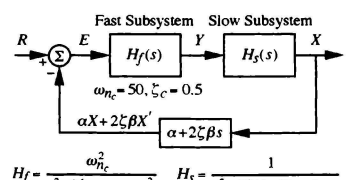


Figure 3. Dual-speed 4th-order system.

transfer function $H(s)$ relating the output X to the input R is given by

$$H(s) = \frac{X(s)}{R(s)} = \frac{1}{s^2 + 2\zeta s + 1} \qquad (8)$$

Here the transfer function represents a 2nd-order system with undamped natural frequency given by $\omega_n = 1$ and a damping ratio given by ζ, regardless of the feedback gain constants α and β. As long as $\omega_{nc}/\omega_n >> 1$, the controller dynamics will have a small effect on the overall system response and Eq. (8) will represent a reasonable approximation to the exact 4th-order system transfer function. To illustrate this we let $\zeta = 0.4$ and $\alpha = \beta = 1$ in Figure 3. Then with $\omega_{nc} = 50$ and $\zeta_c = 0.5$, as shown in the figure, the 4th-order system has two complex conjugate characteristic root pairs. The dominant root pair has an undamped natural frequency given by $\omega_{n_1} = 1.0081$ and a damping ratio $\zeta_1 = 0.39316$, compared with $\omega_n = 1$ and $\zeta = 0.4$ based on Eq. (8). The second root pair exhibits $\omega_{n_2} = 49.598$ and $\zeta_c = 0.39316$, compared with $\omega_{nc} = 50$ and $\zeta_c = 0.5$ for the stand-alone fast subsystem.

We now consider multi-rate simulation of the dual speed system in Figure 3. We recall that multi-rate simulation can be used to improve the computational efficiency in simulating dual-speed systems, with the fast subsystem simulated using an integration frame rate that is a multiple of the frame rate used for the slow subsystem simulation. Figure 4 shows the implementation using an integration step size h (= h_S) for the slow subsystem simulation and an integration step size h_f (= h/N) for the fast subsystem simulation, where N is the integer frame ratio. Extrapolators are used to convert the slow data-sequence output points X_n and X'_n to the fast data-sequence points X_k and X'_k, respectively, required as inputs to the fast-subsystem simulation. The fast-subsystem simulation produces data-sequence output points Y_k, with every Nth data point used as the input Y_n to the slow subsystem simulation.

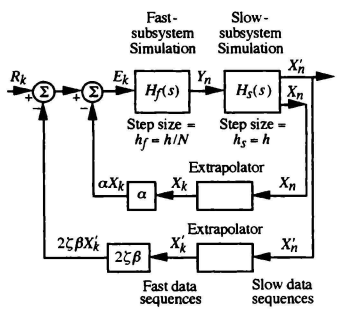


Figure 4. Multi-rate integration of dual-speed system.

When linear extrapolation is used to generate fast data sequence points, the extrapolation formula is given by

$$X_k = X_n + \frac{X_n - X_{n-1}}{h}(t_k - t_n) \quad , \qquad (9)$$

where $t_k - t_n$ is the extrapolation interval. From the appropriate Taylor series expansions it is straightforward to show than the extrapolation error is proportional to $X''_n h^2$.

When quadratic extrapolation is used to generate fast data sequence points, the extrapolation formula is given by

$$X_k = X_n + \frac{3X_n - 4X_{n-1} + X_{n-2}}{2h}(t_k - t_n) + \frac{X_n - 2X_{n-1} + X_{n-2}}{2h^2}(t_k - t_n)^2 \quad . \qquad (10)$$

Again, from the appropriate Taylor series expansions it is straightforward to show that the extrapolation error in this case is proportional to $X'''_n h^3$.

To study the time-domain accuracy of our multi-rate dual-speed system simulation, we will use the same input function employed for the 2nd-order system simulation in the previous section, namely $R(t) = 1 - \cos(t)$. For the parameter values $\zeta = 0.4$, $\alpha = \beta = 1$, $\omega_{nc} = 50$ and $\zeta_c = 0.5$ the response X is quite similar to the 2nd-order system response shown earlier in Figure 1.

As an initial example of dynamic accuracy, we will simulate the dual-speed system using an integration step size given by $h = 0.1$ for the slow-speed system, with $h_f = 0.005$ used for the fast subsystem (frame-ratio $N = 20$). The AB-2 method is again employed for numerical integration, and the quadratic extrapolation of Eq. (10) is used to generate the required multi-rate data sequences. Figure 5 shows the exact simulation output errors, along with the errors calculated from Eq. (3) with $k = 2$, using two solutions, the first with $h_1 = 0.1$ and the second with $h_2 = 0.11$. Both solutions use a frame ratio of $N = 20$. As in the previous section, we see that the experimental method for calculating the dynamic errors yields results which are close to the exact errors.

4. Simulation of the Fast Subsystem Using the State-transition Method

In order to reduce the number of parameters needed

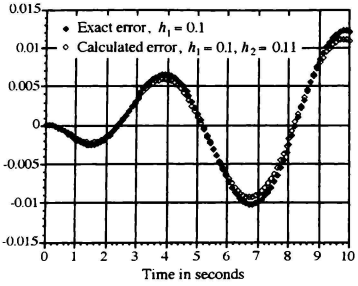


Figure 5. Output error, dual-speed system, $h = 0.1$, quadratic extrapolation, frame-ratio $N = 20$.

to describe our generic multi-rate simulation, we will now employ the state-transition method to simulate the fast subsystem in Figure 3. This is equivalent to using an infinitesimal step size and therefore an infinite frame ratio N. This approach has been shown to give accuracy and stability results that are similar to those obtained with finite frame ratios.[3] In the more practical case where the fast subsystem is nonlinear, the state-transition method can of course not be used, and conventional numerical integration methods with a finite frame ratio N must be employed.

In applying the state-transition method to the fast subsystem, we note that the system response Y can be viewed as the sum of two parts. The first part, $Y_F(t)$, represents the response to the input $R(t)$, whereas the second part, $Y_{fb}(t)$, represents the response to the feedback signal $R_{fb}(t) = -\alpha X(t) - 2\xi\beta X'(t)$. For the input used here, $R(t) = 1 - \cos(t)$, the response $Y_F(t)$ can be obtained using an analytic formula. When quadratic extrapolation is used to generate the slow-to-fast data-sequence conversions in the multi-rate simulation, the response Y_{fb_n} at the discrete time t_n to the feedback signal can be obtained using the following state-transition difference equations[4,5]:

$$Y_{fb_{n+1}} = w_{11}Y_{fb_n} + w_{12}Y'_{fb_n} + R_{fb_n}U(h) + h\hat{R}'_{fb_n}\frac{V(h)}{h} + h^2\hat{R}''_{fb_n}\frac{A(h)}{h^2} \quad , \qquad (11)$$

$$Y'_{fb_{n+1}} = w_{21}Y_{fb_n} + w_{22}Y'_{fb_n} + R_{fb_n}U'(h) + h\hat{R}'_{fb_n}\frac{U(h)}{h} + h^2\hat{R}''_{fb_n}\frac{V(h)}{h^2} \quad . \qquad (12)$$

In Eqs. (11) and (12), $U(h)$, $V($ [illegible] $)$ and $A(h)$ denote the response Y of the 2nd-order syste[illegible] $t = h$ to unit step, unit step-velocity and unit step-acceleration inputs, respectively, at $t = 0$, while w_{11}, w_{12}, w_{21} and w_{22} are the elements of the 2x2 state-transition matrix for the 2nd-order system. The formulas for $U(h), U'(h), V(h), A(h)$, w_{11}, w_{12}, w_{21} and w_{22} are presented in the appendix, along with the formula for Y_{F_n}.

When quadratic extrapolation is utilized for slow-to-fast data-sequence conversion, the following formulas are used for the input derivatives $\hat{R}'_{fb_n}$ and $\hat{R}''_{fb_n}$:

$$\hat{R}'_{fb_n} = -\alpha\hat{X}'_n - \beta(\hat{X}')'_n, \quad \hat{R}''_{fb_n} = -\alpha\hat{X}''_n - \beta(\hat{X}')''_n \quad , \qquad (13)$$

where, from Eq. (10)

$$\hat{X}'_n = \frac{3X_n - 4X_{n-1} + X_{n-2}}{2h} \quad , \quad \hat{X}''_n = \frac{X_n - 2X_{n-1} + X_{n-2}}{h^2} \quad . \qquad (14)$$

Similar formulas are used for $(X'_n)'$ and $(X'_n)''$. When linear extrapolation is utilized for slow-to-fast data-sequence, we let

$$\hat{X}'_n = \frac{X_n - X_{n-1}}{h} \quad , \quad \hat{X}''_n = 0 \quad , \qquad (15)$$

again with similar formulas used for $(X'_n)'$ and $(X'_n)''$.

The error calculations shown in Figure 5 for multi-rate simulation with frame ratio $N = 20$ are now repeated with the state-transition method, corresponding to $N = \infty$, used to emulate multi-rate integration of the fast subsystem. The results are shown in Figure 6. Comparison with Figure 5 shows that replacing multi-rate integration of the fast subsystem for $N = 20$ with the state-transition method produces essentially the same results for both the exact output errors and the approximate errors based on simulations with two different integration step sizes. This confirms the validity of using the

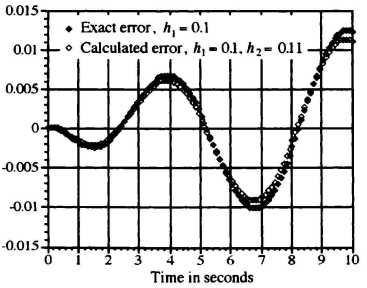


Figure 6. Output error, dual-speed system, $h = 0.1$, quadratic extrapolation, frame-ratio $N = \infty$.

state-transition method to emulate multi-rate simulation of the fast subsystem. We will therefore employ the state-transition method in all remaining examples utilizing the dual-speed system of Figure 3, thereby eliminating the frame ratio N as a parameter in our studies.

We next consider the dual-speed system simulation when quadratic extrapolation is replaced by linear extrapolation, using Eq. (9) for the slow-to-fast data-sequence conversion. For this case Figure 7 compares the exact errors with the calculated errors using Eq. (3), once a-

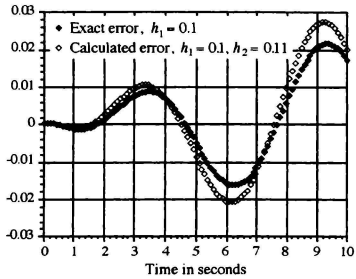


Figure 7. Output error, dual-speed system, $h = 0.1$, linear interpolation, frame-ratio $N = \infty$.

gain with $h_1 = 0.1$ and $h_2 = 0.11$ for the two-solution methodology. Here the calculated errors are somewhat larger than the exact errors. As expected, the overall multi-rate simulation errors are significantly larger than the errors in Figure 6, where the more accurate quadratic extrapolation is used for data-sequence conversion.

5. Effect of a One-frame Time Delay

In creating the program for simulation of dynamic systems it is possible for a time delay of one integration frame to be inadvertently introduced due to a programming error, especially when procedural code such as FORTRAN is utilized. To emulate this in our dual-speed dynamic system, we introduce a delay of h seconds in the input Y to the slow subsystem, where h is the slow-subsystem integration step size. We note that the one-frame delay will introduce dynamic errors proportional to h. For this reason we let $k = 1$ in Eq. (3)for calculating the dynamic error estimate. Figure 8 shows the dynamic errors in this case when $h_1 = 0.1$, $h_2 = 0.11$, along with the exact dynamic error for $h_1 = 0.1$. Also shown is the calculated dynamic error with $k = 2$. The figure shows that the exact dynamic errors fall between the calculated errors for $k = 1$ and $k = 2$. This

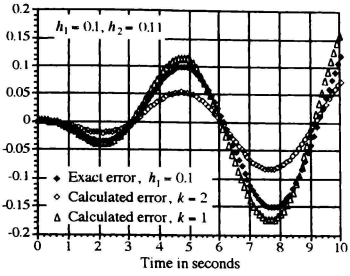


Figure 8. Output error, dual-speed system with one-frame delay in Y, $h = 0.1$, frame-ratio $N = \infty$.

suggests that we utilize a dynamic error model proportional to $C_{1_r}h + C_{2_r}h^2$. To calculate the two proportionality constants we need to run simulations with three different step sizes, namely h_1, h_2 and h_3, with associated data points X_{1_r}, X_{2_r} and X_{3_r}, respectively. Then we can write

$$X_{1_r} - (X_r)_{\text{exact}} \cong C_{1_r}h_1 + C_{2_r}h_1^2 , \quad (16)$$

$$X_{2_r} - (X_r)_{\text{exact}} \cong C_{1_r}h_2 + C_{2_r}h_2^2 , \quad (17)$$

$$X_{3_r} - (X_r)_{\text{exact}} \cong C_{1_r}h_3 + C_{2_r}h_3^2 . \quad (18)$$

Subtracting Eq. (16) from Eqs. (17) and (18) results in two equations, from which the following formulas for the two proportionality constants C_{1_r} and C_{2_r} are obtained:

$$C_{1_r} = \frac{(h_3^2 - h_1^2)(X_{2_r} - X_{1_r}) - (h_2^2 - h_1^2)(X_{3_r} - X_{1_r})}{(h_2 - h_1)(h_3^2 - h_1^2) - (h_3 - h_1)(h_2^2 - h_1^2)} , \quad (19)$$

$$C_{2_r} = \frac{(h_3 - h_1)(X_{2_r} - X_{1_r}) - (h_2 - h_1)(X_{3_r} - X_{1_r})}{-(h_2 - h_1)(h_3^2 - h_1^2) + (h_3 - h_1)(h_2^2 - h_1^2)} . \quad (20)$$

The values of C_{1_r} and C_{2_r} are then used in Eq. (16) to calculate the estimated dynamic error in simulation output.

Using integration step sizes given by $h_1 = 0.1$, $h_2 = 0.11$ and $h_3 = 0.12$, we obtain the dynamic error estimates shown in Figure 9 based on Eqs. (16), (19) and (20) for the case where there is a one-frame delay in the input Y to the slow subsystem. Comparison with the earlier results in Figure 8 for $k = 1$ and $k = 2$ shows that the error model based on $C_{1_r}h + C_{2_r}h^2$, as used in Figure 9, provides a more accurate error estimate than the model based on $C_r h^k$, as used in Figure 10. Note also that the overall simulation errors in Figures 9 and 10 are much larger than the errors in Figures 7 and 8, indi-

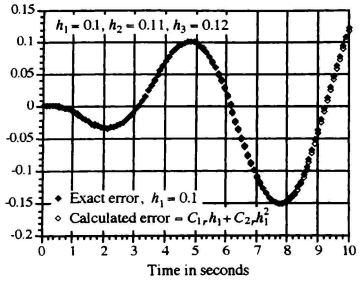


Figure 9. Output error, dual-speed system with one-frame delay in Y, $h = 0.1$, frame-ratio $N = \infty$.

cating the serious decrease in simulation accuracy caused by the one-frame delay.

6. Calculation of the Error Exponent k

All of the dynamic error calculations presented thus far have relied on knowledge of the exponent k associated with the error model used for the calculation. If we assume that the dynamic errors for any given simulation are proportional to h^k, where k can take on non-integer values, then k can be determined numerically from three different simulation runs using integration step sizes h_1, h_2 and h_3, respectively. Thus we can write

$$X_{1_r} - (X_r)_{\text{exact}} \cong C_r h_1^k \,, \tag{21}$$

$$X_{2_r} - (X_r)_{\text{exact}} \cong C_r h_2^k \,, \tag{22}$$

$$X_{3_r} - (X_r)_{\text{exact}} \cong C_r h_3^k \,, \tag{23}$$

from which

$$\frac{X_{3_r} - X_{1_r}}{X_{2_r} - X_{1_r}} = \frac{h_3^k - h_1^k}{h_2^k - h_1^k} \,. \tag{24}$$

From the ratio $(X_{3_r} - X_{1_r})/(X_{2_r} - X_{1_r})$ the exponent k can be calculated numerically at each time t_r. However, to avoid the resulting fluctuations in k it is better to calculate a single k based on the rms (root-mean-square) ratio over the entire simulation interval. Thus we let

$$R(k) = \left[\frac{\sum_{r=1}^{r=M} (X_{3_r} - X_{1_r})^2}{\sum_{r=1}^{r=M} (X_{2_r} - X_{1_r})^2} \right]^{1/2} = \frac{h_3^k - h_1^k}{h_2^k - h_1^k} \,. \tag{25}$$

For a given rms ratio R, the exponent k that satisfies Eq. (25) is calculated iteratively using the formula

$$k_{j+1} = k_j + \frac{R - R(k_j)}{\partial R / \partial k} \,, \tag{26}$$

where k_j is the current trial value of k and k_{j+1} is the next trial value. Here $\partial R / \partial k$ is computed numerically with the formula

$$\partial R / \partial k = \frac{R(k_j + \varepsilon) - R(k_j)}{\varepsilon} \,, \tag{27}$$

where $\varepsilon = 0.00001$.

The exponent k, calculated in this way from the rms ratio R, as obtained from real-time simulations for three different integration sizes, is then used to select the optimal formula for estimating the dynamic errors associated with the simulation. As a first example, consider the case illustrated in Figure 6, where the simulation output errors for $h = 0.1$ are shown for AB-2 integration, with quadratic extrapolation used for slow-to-fast data-sequence conversion and the frame-ratio $N = \infty$ (state-transition method used for the fast subsystem simulation). Table 1 summarizes the rms output errors, as well as the exponent k obtained from Eq. (25), for $h = h_1 = 0.05$, 0.1 and 0.2. For $h_1 = 0.1$, with $h_2 = 0.11$ and $h_3 = 0.12$ the value $k = 1.7258$ suggests that an error calculation based on Ch^2, i.e., Eq. (3) with $k = 2$ should give reasonable results. Indeed, Table 1 shows that the ratio of the rms error calculated in this fashion to the exact rms error is 0.9150, which is consistent with the results shown in Figure 5. Table 1 shows that similar results are obtained when $h_1 = 0.05$ and 0.2, although in each case the error calculation based on $C_2h^2 + C_3h^3$ for some anomolous reason gives better results. Formulas for C_2 and C_3 are given in the appendix.

For the case illustrated in Figure 7, where quadratic extrapolation is replaced by linear extrapolation, Table 1 shows that when $h_1 = 0.1$, the exponent $k = 2.3806$. This suggests that an error calculation based on $C_2h^2 + C_3h^3$ should give superior results, which is confirmed in the table.

For the case illustrated in Figures 8 and 9, where a one frame delay is introduced into the slow-subsystem input Y, Table 2 presents the values of the exponent k obtained from Eq. (25), along with the rms errors resulting from the various error-calculation formulas. For $h_1 = 0.1$ the table shows that the exponent $k = 1.2707$, which suggests that an error calculation based on $C_1h + C_2h^2$ should give the best results. This is confirmed by our previous findings in Figures 8 and 9. The table shows similar results for $h_1 = 0.05$ and 0.2. It should be noted that the value of k observed in each case in Table 2 is considerably less than the value of 2 associated with AB-2 integration errors. This should alert the simulation engineer that there may be a pro-

Table 1

Multi-rate Simulation Output Errors with AB-2 for All Integrations; $\alpha = \beta = 1$; Frame-ratio $N = \infty$

	h_1	h_2	h_3	k_{rms}	Rms output error E_X for integration step size $h = h_1$: $(E_X)_{exact}$	$\frac{(E_X)_{Ch^2}}{(E_X)_{exact}}$	$\frac{(E_X)_{C_1h+C_2h^2}}{(E_X)_{exact}}$	$\frac{(E_X)_{C_2h^2+C_3h^3}}{(E_X)_{exact}}$
Quadratic extrapolation	0.05	0.055	0.06	1.9324	0.001610	0.9815	1.0576	1.0046
	0.1	0.11	0.12	1.7258	0.006162	0.9150	1.2235	1.0045
	0.2	0.22	0.24	1.8061	0.022055	0.8990	1.2755	0.9813
Linear extrapolation	0.05	0.055	0.06	2.2037	0.002312	1.0325	0.8057	0.9597
	0.1	0.11	0.12	2.3806	0.010755	1.2788	0.7147	1.1061
	0.2	0.22	0.24	2.2459	0.051274	1.0545	0.7731	0.9645

Table 2

Multi-rate Simulation Output Errors with AB-2 for All Integrations; $\alpha = \beta = 1$; Frame-ratio $N = \infty$; One-frame Delay in integration of Slow-subsystem input Y

	h_1	h_2	h_3	k_{rms}	Rms output error E_X for integration step size $h = h_1$: $(E_X)_{exact}$	$\frac{(E_X)_{Ch}}{(E_X)_{exact}}$	$\frac{(E_X)_{Ch^2}}{(E_X)_{exact}}$	$\frac{(E_X)_{C_1h+C_2h^2}}{(E_X)_{exact}}$
Quadratic extrapolation	0.05	0.055	0.06	1.1663	0.034678	1.0840	0.5162	0.9953
	0.1	0.11	0.12	1.2707	0.075031	1.1704	0.5573	1.0019
	0.2	0.22	0.24	1.6560	0.176500	1.3986	0.6660	0.9932

gramming error which introduces dynamic errors proportional to h, in this case caused by the one-frame delay in integrating the slow subsystem input Y.

7. Multi-rate Simulation of a Flight Control System

Until now all of our examples of error calculation in multi-rate simulation have utilized the 4th-order dual-speed test problem of Figure 1. In this section we consider the 7th-order dual-speed system shown in Figure 10.[7] It represents an aircraft pitch control system consisting of an airframe, autopilot and control-surface actuator. The flight control system output is the aircraft pitch angle θ, and the input is the command pitch angle, $\theta_i(t)$. Position and rate feedback are used along with θ_i as autopilot inputs. For simplicity, the dynamic behavior of the airframe is modeled as a third-order linear system. Utilizing a full nonlinear model of the airframe would not alter significantly the simulation results presented here. The dynamic behavior of the airframe is dominated by ω_{n_s}, the undamped natural frequency of the short-period pitching motion. In our example simulation we will let $\omega_{n_s} = 5$ rad/sec, which is representative of an agile fighter aircraft.

The control surface actuator is modeled as the closed-loop system shown in Figure 11, i.e., a pure inertia plant driven by a proportional-plus-rate controller with a double lag. The actual parameters used in the example simulation are given by $K_{act} = 500$, $C_e = 0.08$ and $\tau = 0.01$. The remaining parameters in Figure 10 are given by $K = 0.1625$ and $C_d = 0.2$. The overall flight control system then exhibits closed-loop eigenvalues given by $-4.31 \pm j6.418$, -0.6729, $-11.067 \pm j37.453$, -16.114 and -150.411, where the first three roots are associated with the slow airframe subsystem and the last four roots are associated with the fast actuator subsystem.

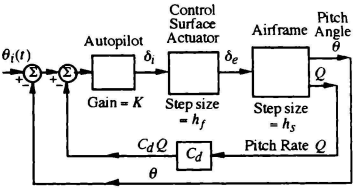


Figure 10. Example flight control system.

We will employ the acceleration-limited step function shown in Figure 12 as the test input θ_i to the

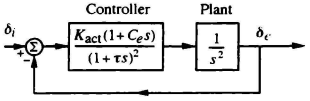


Figure 11. Control surface servo actuator.

flight control system. Also shown in the figure are the airframe and actuator responses, as computed using RK-4 integration with a step size of 0.00225 seconds. This simulation is accurate enough to serve as a reference against which subsequent real-time simulations can be checked to obtain exact dynamic errors. With the input acceleration reversal time of 0.05 seconds used here, the airframe output θ and actuator output δ_e are very close to their respective outputs for an ideal step input. By using the acceleration-limited step input, we avoid the overwhelmingly large dynamic errors associated with real-time predictor methods when integrating inputs with discontinuities in displacement or slope.

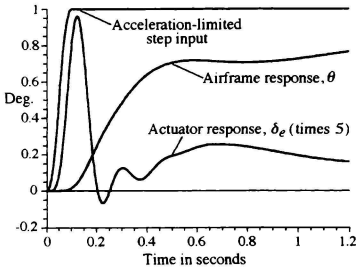


Figure 12. Response of flight control system to acceleration-limited step input.

We now consider real-time simulation of the flight-control system of Figure 10 using multi-rate AB-2 integration with a step size h_s (= h) for the airframe subsystem and a step size h_f (= h_s/N) for the actuator subsystem. In selecting integration step sizes, we must bear in mind that the maximum possible step size will be governed by the eigenvalue of largest magnitude for the subsystem being simulated which, for the actuator subsystem, is equal to -149.5. In the case of negative real eigenvalues λ, it is straightforward to show for AB-2 integration that $|\lambda h|$ must be less than 1 for the simulation to be stable. For $\lambda = -149.5$ it follows that h_f must be less than 0.00669. Based on this, we let h_f = 0.0045 for our initial AB-2 simulation. We will let the frame ratio $N = 4$, so that the step size for the airframe simulation is given by $h = 0.018$. As in our previous examples, we use quadratic extrapolation to convert the slow data-sequence outputs from the airframe simulation to the required fast data-sequence inputs to the autopilot-actuator simulation. Figure 13 shows the resulting simulation errors based on the RK-4 reference solution with $h = 0.00225$, which is taken to be exact. To avoid clutter, only every fourth data point is shown for the actuator output errors.

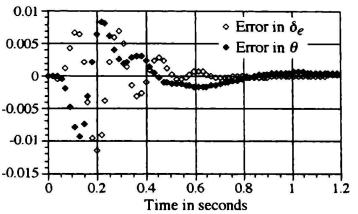


Figure 13. Output errors in simulation of multi-rate flight control system; $h = 0.018$, $h_f = 0.0045$ ($N = 4$).

We now consider the experimental calculation of dynamic errors in simulating the7th-order flight control system. As before we utilize three different integration step sizes in order to compute the dependence of dynamic error on step size. The results are summarized in Table 3. The value of k_{rms} that satisfies the rms error ratio defined in Eq. (25) is listed for each case in the table. Thus for airframe integration step sizes of 0.009, 0.010 and 0.011 (the corresponding actuator step sizes are 0.00225, 0.00250 and 0.00275, respectively), we find that the dynamic errors in θ vary as $h^{2.1004}$. Hence it is not surprising that utilizing Eq. (1) with $k = 2$ exhibits the best agreement with the exact rms error for h = 0.009. On the other hand, for integration step sizes of 0.018, 0.020 and 0.022, Table 3 shows that the dynamic errors in θ vary as $h^{2.6106}$. In this case calculating the error based on $C_2h^2 + C_3h^3$ gives good agreement with the exact rms error for $h = 0.018$.

In Table 3 we see that for airframe integration step sizes of 0.009, 0.010 and 0.011 (the corresponding actuator step sizes are 0.00225, 0.00250 and 0.00275, respectively), the dynamic errors in actuator output vary as $h^{2.306}$. This suggests that calculating the error based on $C_2h^2 + C_3h^3$ should give the best agreement with the exact rms error, which is indeed the case. On the other hand, for integrator step sizes of 0.018, 0.020 and 0.022 (the corresponding actuator step sizes are 0.0045, 0.005

and 0.055, respectively), Table 3 indicates that the dynamic errors in δ_e vary as $h^{3.4441}$. In this case an error calculation based on $C_3h^3 + C_4h^4$ gives the best results, with $[(E)_{C_3h^3+C_4h^4}]/(E)_{\text{exact}} = 0.8957$.

We now turn to consideration of the effect of introducing a one-frame time delay in the input δ_e to the slow airframe system. As pointed out in Section 5, the inadvertent introduction of such a delay can occur when programming real-time simulations. Table 4 summarizes the output error calculations in this case. For airframe integration step sizes of 0.009, 0.010 and 0.011, Table 4 shows that the dynamic errors in airframe output θ vary as $h^{1.3667}$, compared with $h^{2.306}$ in Table 3 for no delay. In this case we expect the best results with the error calculation based on $C_1h + C_2h^2$, which the table confirms. We have already noted in Section 5 that the time delay introduces dynamic errors directly proportional to the delay h, which accounts for the substantially lower exponent k in Table 4, as well as the much larger rms error. For integration step sizes of 0.018, 0.020 and 0.022, Table 4 shows that the dynamic errors in θ vary as $h^{2.0088}$. This suggests that the most accurate results will be obtained when the error calculation is based on Ch^2, whereas the table shows the most accurate results when the error is based on $C_1h + C_2h^2$. The author believes that the explanation for this inconsistency lies in the fact that when $h = 0.018$ and a one-frame delay is also present, the amplitude of the dynamic errors is no longer small compared with the exact transient solution amplitude, in which case all of the formulas introduced here for experimental error calculations become suspect.

In Table 4 we see that for airframe integration step sizes of 0.009, 0.010 and 0.011 (the corresponding actuator step sizes are 0.00225, 0.00250 and 0.00275, respectively), and a one-frame time delay, the dynamic errors in actuator output δ_e vary as $h^{1.3553}$. In this case we expect the error calculation based on $C_1h + C_2h^2$ to give the best results, which the table confirms. For airframe step sizes of 0.018, 0.020 and 0.022, i.e., actuator step sizes of 0.0045, 0.005 and 0.055, respectively), Table 4 indicates that the dynamic error in δ_e varies as $h^{2.2751}$. This suggests a calculation with the error based on $C_2h^2 + C_3h^3$ should be the most accurate, which the table shows is marginally true. Again, the explanation for this probably lies in the fact that for

Table 3

Flight Control System Simulation Output Errors Using Quadratic Extrapolation with Multi-rate AB-2 Integration; Frame-ratio $N = 4$

					Rms output error E for integration step size $h = h_1$			
	h_1	h_2	h_3	k_{rms}	$(E)_{\text{exact}}$	$\frac{(E)_{Ch^2}}{(E)_{\text{exact}}}$	$\frac{(E)_{C_1h+C_2h^2}}{(E)_{\text{exact}}}$	$\frac{(E)_{C_2h^2+C_3h^3}}{(E)_{\text{exact}}}$
Airframe output errors	0.009	0.010	0.011	2.1004	0.000705	1.0190	2.1512	1.0981
	0.018	0.020	0.022	2.6106	0.002929	1.1470	1.6703	1.0330
Actuator output errors	0.009	0.010	0.011	2.3060	0.000635	1.0671	0.7565	0.9564
	0.018	0.020	0.022	3.4441	0.003032	1.3351	3.5817	1.2374

Note: for $h = 0.018$, $\dfrac{(E_{\delta_e})_{C_3h^3+C_4h^4}}{(E_{\delta_e})_{\text{exact}}} = 0.8957$

Table 4

Flight Control System Simulation Output Errors Using Quadratic Extrapolation with Multi-rate AB-2 Integration and a One Slow-frame Time delay in δ_e; Frame-ratio $N = 4$

					Rms output error E for integration step size $h = h_1$				
	h_1	h_2	h_3	k_{rms}	$(E)_{\text{exact}}$	$\frac{(E)_{Ch}}{(E)_{\text{exact}}}$	$\frac{(E)_{Ch^2}}{(E)_{\text{exact}}}$	$\frac{(E)_{C_1h+C_2h^2}}{(E)_{\text{exact}}}$	$\frac{(E)_{C_2h^2+C_3h^3}}{(E)_{\text{exact}}}$
Airframe output errors	0.009	0.010	0.011	1.3667	0.006360	1.1685	0.5535	0.9667	0.6752
	0.018	0.020	0.022	2.0088	0.015064	0.7109	1.5008	1.1239	0.7591
Actuator output errors	0.009	0.010	0.011	1.3553	0.005133	1.1452	0.5425	0.9546	0.6639
	0.018	0.020	0.022	2.2751	0.012041	1.5392	0.7291	1.2442	0.7502

these large step sizes, accompanied by the one-frame time delay, the dynamic errors in actuator output are no longer small compared with the exact transient solution amplitude. Thus the rms error in Table 4, $(E)_{\text{exact}} = 0.012041$, is 21 percent of the entire rms actuator response shown in Figure 13.

8. Error Calculation for a Hardware-in-the-Loop Simulation

We next consider the real-time simulation of our flight-control system when the actuator and autopilot are replaced with the actual flight hardware. In this case a single processor is used to simulate the airframe. To investigate the effectiveness of our on-line methodology for calculating the dynamic errors in the real-time computer simulation, we will emulate the continuous hardware system by using a very small integration step size, in order to approach closely the actual performance of the continuous system. Thus we let the step size for the actuator simulation be given by $h_j = h/18$, where h is the integration step size for the airframe simulation. With the frame ratio N thus equal to 18, it turns out that the simulation of the actuator exhibits negligible dynamic errors.

In our hardware-in-the-loop simulation, it is also necessary to emulate the interfaces between the autopilot-actuator hardware and the processor that simulates the airframe. The A to D interface between the actuator hardware and the airframe simulation will be emulated by simply letting $\delta e_n = \delta e_k$ whenever $t_n = t_k$, where t_n is the discrete time associated with the airframe simulation data points and t_k is the discrete time associated with the autopilot-actuator emulation data points. We will assume that the D to A interface for converting the digital output data sequences from the airframe simulation to the continuous inputs for the autopilot-actuator hardware utilizes zero-order extrapolation. We will also assume that the D to A converters are driven at the same frame rate as the airframe simulation. Note that the step discontinuities in the zero-order-hold outputs of the D to A converters will cause a large error when integrated using the AB-2 predictor method for the autopilot-actuator emulation. For this reason, we integrate the DAC (D to A converter) outputs using the Euler formula. All other integrations are performed using the AB-2 method.

Table 5 summarizes the simulation output errors for the hardware-in-the-loop emulation described above. For integration step sizes of 0.009, 0.010 and 0.011, the dynamic errors vary as $h^{1.2947}$. This suggests the most accurate results for error calculations based on $C_1 h + C_2 h^2$, which the table confirms. For integration step sizes of 0.018, 0.020 and 0.022, the dynamic errors vary as $h^{1.7099}$. Again, this suggests the most accurate results for error calculations based on $C_1 h + C_2 h^2$, which once more the table confirms. It should be noted that use of a DAC with zero-order hold is equivalent to introducing a half-frame time delay into the simulation.[6] This in turn produces dynamic errors proportional to h. With the dynamic errors produced by AB-2 integration proportional to h^2, this explains why the exponent k of the error dependence in Table 5 falls between 1 and 2, with k nearer to 2 for the larger step sizes. Not shown in Table 5 is the result of an error calculation based on Ch^k, where $k = 1.7099$. In this case $[(E)_{Ch^k}]/(E)_{\text{exact}} = 0.8783$, a result that is inferior to the calculation based on $C_1 h + C_2 h^2$.

The effect of the one-half frame time delay introduced by the DAC with zero-order hold can be eliminated by advancing the DAC input data points by one-half frame using quadratic extrapolation. The dynamic error then produced by the DAC with zero-order hold can be shown to be proportional to h^2.[6] Table 5 summarizes the simulation output errors in this case. For integration step sizes of 0.009, 0.010 and 0.011, the dynamic errors now vary as $h^{1.9409}$. This suggests the most accurate results for error calculations based on Ch^2, which the table confirms. For integration step sizes of 0.018, 0.020 and 0.022, the dynamic errors vary as $h^{2.4885}$. This suggests the most accurate results for error calculations based on $C_2 h^2 + C_3 h^3$, which again is confirmed in Table 5.

In all of the hardware-in-the-loop simulations considered thus far we have assumed that the D to A conversion of the airframe simulation outputs takes place at a conversion rate equal to the integration frame rate of the airframe simulation. It is apparent that the dynamic errors introduced by the DACs can be reduced by using a conversion rate which is a multiple of the frame rate of the digital data sequences being converted to analog form. For example, let us assume that we drive the DACs with data sequences which have three times the frame rate of the airframe output data sequences. This means that the step size h_{DAC} associated with the D to A conversion process is equal to $h/3$, where h is the time step used for the airframe simulation. To generate the required multi-rate DAC input data sequences we can employ the same quadratic extrapolation formulas used for slow-to-fast data sequence conversion in our the earlier multi-rate simulations. When this is done, the results summarized in Table 6 are obtained for the estimated dynamic errors in the simulated airframe outputs, where as before we emulate the autopilot-actuator hardware using multi-rate simulation with the frame-rate ratio $N = 18$. Thus in Table 6 for airframe integration step sizes of 0.009, 0.010 and 0.011 we find that the

Table 5

Simulation Errors in Airframe Output Using Quadratic Extrapolation with Emulation of Actuator Hardare; $h_{DAC} = h$; Frame-ratio $N = 18$

					Rms output error E for integration step size $h = h_1$				
	h_1	h_2	h_3	k_{rms}	$(E)_{exact}$	$\frac{(E)_{Ch}}{(E)_{exact}}$	$\frac{(E)_{Ch^2}}{(E)_{exact}}$	$\frac{(E)_{C_1h+C_2h^2}}{(E)_{exact}}$	$\frac{(E)_{C_2h^2+C_3h^3}}{(E)_{exact}}$
No DAC lead	0.009	0.010	0.011	1.2947	0.004771	1.1552	0.5472	0.9965	0.6802
	0.018	0.020	0.022	1.7099	0.010988	1.3982	0.6623	1.0883	0.7599
$h/2$ DAC lead	0.009	0.010	0.011	1.9409	0.000702	2.1425	1.0149	1.1138	1.0389
	0.018	0.020	0.022	2.4885	0.002885	2.3947	1.1344	1.6645	1.0622

Table 6

Simulation Errors in Airframe Output Using Quadratic Extrapolation with Emulation of Actuator Hardare; $h_{DAC} = h/3$; Frame-ratio $N = 18$

					Rms output error E for integration step size $h = h_1$				
	h_1	h_2	h_3	k_{rms}	$(E)_{exact}$	$\frac{(E)_{Ch}}{(E)_{exact}}$	$\frac{(E)_{Ch^2}}{(E)_{exact}}$	$\frac{(E)_{C_1h+C_2h^2}}{(E)_{exact}}$	$\frac{(E)_{C_2h^2+C_3h^3}}{(E)_{exact}}$
No DAC lead	0.009	0.010	0.011	1.5602	0.001923	1.3817	0.6545	1.0105	0.7570
	0.018	0.020	0.022	2.1688	0.005236	1.8057	0.8553	1.3046	0.8772
$h/6$ DAC lead	0.009	0.010	0.011	1.9586	0.000707	2.1479	1.0174	1.0958	1.0352
	0.018	0.020	0.022	2.6651	0.002943	2.4741	1.1720	1.9097	1.0717

dynamic errors in θ vary as $h^{1.5602}$. This suggests that utilizing Eq. (22), with the error based on $C_1h + C_2h^2$ will yield the most accurate results, which the table confirms. For airframe integration step sizes of 0.018, 0.020 and 0.022, the dynamic errors vary as $h^{2.1688}$. This suggests the most accurate results for error calculations based on $C_2h^2 + C_3h^3$, which again is corroborated in Table 6. For the case of no DAC lead, comparison of the rms errors, $(E)_{exact}$, in Table 6 with the rms errors in Table 5 confirms that using multi-rate D to A conversion for the hardware-in-the-loop simulation indeed reduces the simulation output errors.

The multi-rate DAC results are repeated in Table 6 for the case where the half-frame delay associated with the DAC outputs is eliminated by advancing the DAC input data points by $h_{DAC}/2$, i.e., $h/6$, using quadratic extrapolation. For airframe integration step sizes of 0.009, 0.010 and 0.011, the dynamic errors vary as $h^{1.9586}$. This suggests the most accurate results for error calculations based on Ch^2, which is confirmed in Table 6. For airframe integration step sizes of 0.018, 0.020 and 0.022, the dynamic errors vary as $h^{2.6651}$. In this case the most accurate results should be obtained when the error is based on $C_2h^2 + C_3h^3$, which the table shows to be true. It is also interesting to note that the rms errors, $(E)_{exact}$, in Table 6 for $h/6$ DAC lead are no less than the corresponding rms errors for $h/2$ DAC lead in Table 5. This indicates that when a half-frame lead is utilized in driving the DACs, no further accuracy improvement is obtained by driving the DACs with a multi-rate data sequence, at least in this example.

There is one potential difficulty associated with the calculation of real-time dynamic errors when actual hardware is involved, namely, the lack of exact run-to-run repeatability. Only for the case where the hardware takes the form of a digital processor will exact repeatability occur. Otherwise, the effect of random noise in hardware subsystems, including D to A and A to D converters, may very likely prevent accurate on-line calculation of dynamic errors using two different simulation step sizes. This is because the various formulas we have utilized to compute the dynamic errors all rely on the small difference between solutions for different integration step sizes. The accuracy of these small difference computations may very well be compromised by the lack of repeatability in the hardware, which will then result in non-repeatable error calculations of unacceptable reliability. However, a way around this problem is to include in the initial simulation run of step size h_1 a logging in digital form of all hardware outputs that in turn provide inputs to the computer portion of the HITL simulation. The logged

hardware output data for the initial run of step size h_1 is then used to provide inputs to the computer portion of the simulation for subsequent runs with different step sizes, with accurate interpolation formulas used to correct any time mismatches between logged hardware output data points and required computer input data points. By using exactly the same data as computer inputs for the simulation runs with different integration step sizes, we ensure exactly repeatable results. It should be noted that this procedure effectively breaks the feedback loop between computers and hardware for simulation runs other than the original run with step size h_1. This in turn means that the errors calculated using the on-line formulas will represent open-loop dynamic errors associated with the computer simulation with integration step size h_1 rather than closed-loop dynamic errors.

To illustrate the technique, we again consider the HITL simulation of the flight-control system of Figure 10, with the nominal airframe simulation step size given by $h_S = h = 0.009$ and the actuator hardware emulated using a step size given by $h_f = h_S/18 = 0.0005$. The D to A conversion is emulated as described at the beginning of this section, with $h_{DAC} = h = 0.009$. The time history of the actuator output data points δ_e is recorded and subsequently played back as airframe inputs for second and third simulations with step sizes given by $h = h_2$ and h_3, respectively.

In Figure 14 the errors in airframe output, as calculated using Eq. (3) with $k = 2$, $h_1 = 0.009$ and $h_2 = 0.010$ are compared with the exact errors for $h = 0.009$. Here the exact errors are based on an RK-4 simulation of the airframe with $h = 0.0005$, where the actuator output data points, as obtained from the closed-loop simulation described above, are used as airframe simulation

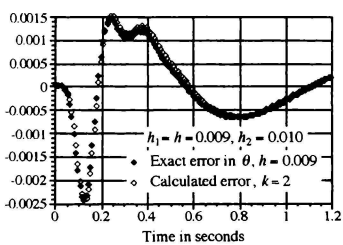


Figure 14. Open-loop simulation errors in airframe output using as input data the actuator output from a closed-loop simulation with $h_S = 0.009$, $h_f = 0.0005$.

inputs. Since the only dynamic errors in the open-loop airframe simulation are those caused by the AB-2 integration method, it is not surprising that the estimated errors using Eq. (3) with dynamic errors proportional to h^2, are close to the exact errors.

In Table 7 the results based on various methods for estimating the airframe simulation errors using different integration step sizes are summarized. For the case illustrated in Figure 14, with $h_1 = 0.009$, $h_2 = 0.010$ and $h_3 = 0.011$, the exponent k, calculated from the rms ratio R defined in Eq. (25), is equal to 1.8354. Table 7 shows that when the estimated error for $h = 0.009$ is based on Ch^2, the resulting rms error is 1.0351 times the exact rms error, with all other schemes for calculating the estimated error yielding inferior results. For $h_1 = 0.018$, $h_2 = 0.020$ and $h_3 = 0.022$, Table 7 shows that $k = 2.0224$, which again suggests that the most accurate results will be obtained when the estimated error is based on Ch^2, which the table verifies.

9. Summary

In this paper we have used three different dynamic systems to illustrate the methodology for experimental calculation of dynamic errors in real-time simulation. The dynamic error calculation is based on comparing two real-time simulations, one with integration step size h_1 and output data points X_{1r} at the discrete times t_r, and the second with an augmented step size h_2 and output data points X_{2n} at the discrete times t_n. Accurate interpolation formulas are then used to convert the outputs from the second simulation to data points X_{2r} at the discrete times t_r for the first simulation, from which the difference between the two simulations, $X_{2r} - X_{1r}$ at common discrete times t_r can be calculated. Assuming that the dynamic errors are proportional to h^k, Eq. (3) can then be used to calculate directly the dynamic errors for the simulation with integration step size h_1.

The exponent k can be determined experimentally by running an additional simulation with step size h_3 and calculating the difference $X_{3r} - X_{1r}$ between the third and first simulations. The rms ratio R given in Eq. (25) is then calculated from the mean square differences $X_{3r} - X_{1r}$ and $X_{2r} - X_{1r}$ from which k can be computed. We have found from numerous examples that Eq. (3) gives reliable dynamic error calculations only when k is set equal to an integer. Thus when the value of k obtained from Eq. (25) is close to an integer value, that integer value is used in Eq. (3) to calculate the dynamic error. On the other hand, when k based on Eq. (25) falls roughly midway between two integer values, a dynamic error based on both the integer values is found to give

Table 7

Open-loop Simulation Errors in Airframe Output Using as Input Data the Actuator Output from a Closed-loop Simulation with $h_s = h_1 = 0.009$, $h_f = 0.00005$.

				Rms output error E for integration step size $h = h_1$				
h_1	h_2	h_3	k_{rms}	$(E)_{exact}$	$\frac{(E)_{Ch}}{(E)_{exact}}$	$\frac{(E)_{Ch^2}}{(E)_{exact}}$	$\frac{(E)_{C_1h+C_2h^2}}{(E)_{exact}}$	$\frac{(E)_{C_2h^2+C_3h^3}}{(E)_{exact}}$
0.009	0.010	0.011	1.8354	0.000824	2.1853	1.0351	1.4584	1.1281
0.018	0.020	0.022	2.0224	0.003467	2.3051	1.0919	2.0510	1.2166

the most reliable results. For example, when the k value obtained from Eq. (25) is roughly half-way between 1 and 2, Eq. (16) based on an error given by $C_1h + C_2h^2$ gives the most accurate dynamic error calculation. On the other hand, when the k value obtained from Eq. (25) is roughly half-way between 2 and 3, an error based on $C_2h^2 + C_3h^3$ gives the most accurate calculation.

The methodology described above for calculating dynamic errors in real-time simulation has been illustrated using the 4th-order dual-speed system of Figure 3 with multi-rate frame ratios given by $N = 20$ and $N = \infty$. Cases considered include the use of either linear or quadratic extrapolation for the required slow-to-fast data-sequence conversion in the multi-rate simulations. Also considered is the case where a one-frame time delay is inadvertently introduced into the simulation.

The methodology has also been illustrated using the 7th-order flight-control system of Figure 10 with multi-rate frame ratio given by $N = 4$. To emulate the case where computer simulation of the fast subsystem is replaced by the actual hardware, the hardware is simulated with the very small step size that results when $N = 18$. In addition, the D to A conversion of the slow-subsystem outputs is emulated, both for the case where the half-frame delay associated with the D to A conversion is uncompensated, and compensated using quadratic extrapolation. To eliminate the unreliable dynamic error calculations that result from lack of exact run-to run repeatability in hardware-in-the-loop simulations, the hardware outputs in digital form produced by the first simulation with nominal step size h_1 are logged and subsequently played back as computer inputs for additional simulations with step sizes h_2 and h_3. The resulting error calculations then represent the open-loop dynamic errors associated with the computer portion of the hardware-in-the-loop simulation.

The procedure of first determining the exponent k in the dynamic error variation as h^k based on simulation runs with three different integration step sizes, followed by calculation of the dynamic error itself using the appropriate formulas, as described above, has been shown to give reasonably reliable results in the many examples considered in this paper. The procedure also permits the detection of undesirable time delays in the overall simulation, as identified when the calculated exponent k is found to be close to 1. The author feels that the methodology introduced here can play a major role in optimizing real-time, hardware-in-the-loop simulations, and in verifying the dynamic fidelity of such simulations.

References

1. Howe, R.M., "Real-time Simulation of Dynamic Systems Using a Variable Integration Step Size," *Proc. of the 1993 Simulation Multiconference*, Arlington, VA, March 29-April 1, 1993, pp 452-458.

2. Deiss, W.H., E.J. Fadden and R.M. Howe,"An Asynchronous Simulation Methodology for Improving the Performance of Training Simulators," *Proc. of the AIAA Flight Simulation Technologies Conference*, Monterey, CA, Aug. 10-12, 1993.

3. Laffitte, J., and R.M. Howe., "Interfacing Fast and Slow Subsystems in the Real-time Simulation of Dynamic Systems," *Transactions of the Society for Computer Simulation*, Vol. 14, No. 3, September 1997, pp. 115-126.

4. Chen, C.T., *Introduction to Linear System Theory*, Holt, Reinhart and Winston, Inc., 1960.

5. Howe, R.M., *Dynamics of Real-time Simulation*, Applied Dynamics International, 3800 Stone School Road, Ann Arbor, Michigan 48108, September, 1998.

6. Howe, R.M., "Dynamics of Digital Interpolation and Extrapolation," *Transactions of the Society for Computer Simulation*, Vol. 2, No. 2, June 1985, pp. 169-187.

7. Howe, R.M., "On-line Calculation of Dynamic Errors in Real-time Simulation," *Proc. of SPIE Conference on Enabling Technology for Simulation Science II*, Orlando, FL, April, 1998, pp 319-330.

Appendix

A.1 Coefficients for State-transition Simulation of Underdamped 2nd-order System

Listed below are the formulas for the 2x2 matrix elements in Eqs. (11) and (12) for simulating an underdamped ($\zeta < 1$) 2nd-order system with undamped natural frequency ω_n and damping ratio ζ, as well as the formulas for the system response at the time $t = h$ to unit step, unit-step velocity, unit-step acceleration and unit-im-pulse inputs applied at $t = 0$:

$$w_{11} = e^{-\zeta\omega_n h}\left[\cos(\omega_d h) + \frac{\zeta\omega_n}{\omega_d}\sin(\omega_d h)\right],$$

$$w_{12} = \frac{1}{\omega_d}e^{-\zeta\omega_n h}\sin(\omega_d h), \quad w_{21} = -\frac{\omega_n^2}{\omega_d}e^{-\zeta\omega_n h}\sin(\omega_d h),$$

$$w_{22} = e^{-\zeta\omega_n h}\left[\cos(\omega_d h) - \frac{\zeta\omega_n}{\omega_d}\sin(\omega_d h)\right], \quad \text{(A.1)}$$

$$U(h) = 1 - w_{11}, \quad \frac{V(h)}{h} = 1 - \frac{2\zeta}{\omega_n h} +$$

$$\frac{2\zeta}{\omega_n h}e^{-\zeta\omega_n h}\left[\cos(\omega_d h) - \frac{(1-2\zeta^2)\omega_n}{\omega_d}\sin(\omega_d h)\right],$$

$$\frac{A(h)}{h^2} = \frac{4\zeta^2-1}{\omega_n^2 h^2} - \frac{2\zeta}{\omega_n h} + \frac{1}{2} +$$

$$\frac{e^{-\zeta\omega_n h}}{\omega_n^2 h^2}\left[(1-4\zeta^2)\cos(\omega_d h) + \frac{(3\zeta - 4\zeta^3)\omega_n}{\omega_d}\sin(\omega_d h)\right],$$

$$U(h) = \omega_n^2 w_{12}, \quad \omega_d = \omega_n\sqrt{1-\zeta^2} \quad \text{(A.2)}$$

A.2. Formula for Response of 2nd-order System to 1-cos(ωt) Input

Listed below are the formulas for the response of a 2nd-order system with undamped natural frequency ω_n and damping ratio ζ to the input 1 - cos(ωt), zero initial conditions:

$$Y_{r_n} = 1 - M\cos(\omega nh + N) +$$

$$e^{-\zeta\omega_n nh}[A\cos(\omega_d nh) + B\sin(\omega_d nh)], \quad \text{(A.3)}$$

where

$$M = \left[(1-\omega^2/\omega_n^2)^2 + (2\zeta\omega k\omega_n)^2\right]^{-1/2},$$

$$N = \tan^{-1}\left[\frac{-2\zeta\omega k\omega_n}{1-\omega^2/\omega_n^2}\right], \quad A = M\cos N - 1,$$

$$B = (\zeta\omega_n A - \omega M\sin N)/\omega_d, \quad \omega_d = (1-\zeta^2)^{1/2}. \quad \text{(A.4)}$$

A.3. Formulas for Coefficients C_2 and C_3

For a dynamic error model given by $C_{2_r}h^2 + C_{3_r}h^3$ and simulation output data points X_{1_r}, X_{2_r} and X_{3_r}, as obtained with integration step sizes h_1, h_2 and h_3, respectively, we can write

$$X_{1_r} - (X_{1_r})_{\text{exact}} \cong C_{2_r}h_1^2 + C_{3_r}h_1^3, \quad \text{(A.5)}$$

$$X_{2_r} - (X_{2_r})_{\text{exact}} \cong C_{2_r}h_2^2 + C_{3_r}h_2^3, \quad \text{(A.6)}$$

$$X_{3_r} - (X_{3_r})_{\text{exact}} \cong C_{2_r}h_3^2 + C_{3_r}h_3^3. \quad \text{(A.7)}$$

Subtracting Eq. (A.5) from Eqs. (A.6) and (A.7) results in two equations, from which the following formulas for the two proportionality constants are obtained:

$$C_{2_r} = \frac{(h_3^3-h_1^3)(X_{2_r}-X_{1_r}) - (h_2^3-h_1^3)(X_{3_r}-X_{1_r})}{(h_2^2-h_1^2)(h_3^3-h_1^3) - (h_3^2-h_1^2)(h_2^3-h_1^3)}, \quad \text{(A.8)}$$

$$C_{3_r} = \frac{(h_3^2-h_1^2)(X_{2_r}-X_{1_r}) - (h_2^2-h_1^2)(X_{3_r}-X_{1_r})}{-(h_2^2-h_1^2)(h_3^3-h_1^3) + (h_3^2-h_1^2)(h_2^3-h_1^3)}. \quad \text{(A.9)}$$

A.4. Formulas for Coefficients C_3 and C_4

To derive the corfficient formulas for a dynamic error model given by $C_{3_r}h^3 + C_{4_r}h^4$, we follow the same procedure used above in Section A.3. Thus we obtain

$$C_{3_r} = \frac{(h_3^4-h_1^4)(X_{2_r}-X_{1_r}) - (h_2^4-h_1^4)(X_{3_r}-X_{1_r})}{(h_2^3-h_1^3)(h_3^4-h_1^4) - (h_3^3-h_1^3)(h_2^4-h_1^4)}, \quad \text{(A.10)}$$

$$C_{4_r} = \frac{(h_3^3-h_1^3)(X_{2_r}-X_{1_r}) - (h_2^3-h_1^3)(X_{3_r}-X_{1_r})}{-(h_2^3-h_1^3)(h_3^4-h_1^4) + (h_3^3-h_1^3)(h_2^4-h_1^4)}. \quad \text{(A.11)}$$

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37269

AIAA-2000-4395

VECTORIZATION BLOCKS FOR SIMULINK

Giampiero Campa‡, Mario Innocenti†,
Department of Electrical Systems and Automation, University of Pisa, 56126 Pisa, ITALY

Abstract

The paper presents two blocks that extend Simulink range of applicability to varying matrices, allowing the user to build (without writing a single line of code) dynamical systems that can be both complex and suitable for real time simulations. First of all, the three common techniques to build dynamical system with Simulink will be presented, showing the advantages and disadvantages of each of them. Then the explanation is given as why (when dealing with complex multivariable systems) neither of the three can be at the same time easy to use and suitable for real time simulations. Two Simulink blocks will then be proposed that solve this problem by allowing varying matrices to be handled directly in the Simulink environment. The aspect, the user interface, and the most important part of the code of these blocks will be discussed, and their use will finally be explained by means of some examples. In particular, we will present an example on how to implement a complex multilayer adaptive neural network using these two blocks.

Introduction: S-functions

S-functions are the way in which a dynamical system is represented inside Simulink environment. Every Simulink object (block, composition of blocks, schemes) is a dynamical system; hence, every object (included the entire simulation scheme) is represented by an S-function.

Typically, the built in Simulink blocks are associated with precompiled S-functions, while user added S-functions are obtained joining blocks or directly written in a language such as Matlab, C, C++, Fortran.

Whatever the language in which the S-function is written in, it must interact with Simulink in the same way, that is, it must have the structure shown in [1].

Different ways to build S-functions

Joining together Simulink blocks

Perhaps the most common way to build dynamical systems (hence S-functions) in Simulink, is just to join together Simulink blocks. For example, if we have two blocks that represent two systems and we would like to have the system consisting of the series of the two, the easiest way to obtain the series system is just drawing a wire that connects the output of the first system to the input of the second one.

We can then group the two systems together using the Simulink "Group" function in order to obtain a block that appears as any Simulink block and represents our "new" dynamical system.

The ease of operations like the one depicted above has been maybe the main cause of Simulink success as an interface to deal with dynamical systems, indeed in this way we can build a dynamical system straightforwardly without the need to write a single line of code.

In some cases however assembling subsystems may not be the best choice, this is true especially if we already have the input-state-output equations of the system that we are going to build, and especially for nonlinear multivariable systems.

Let's consider for example a system in which the output depends from the state vector in this way:

$$y = C(x)D(x)x$$

where each entry of the matrices C and D is a given function of x. If we want to obtain y by connecting some blocks that transform the signals x into y, then the only solution is to do the multiplication in a wire by wire fashion, if both C and D are 2 by 2 matrices we have:

$$\begin{aligned} y_1 &= (C_{11}(x)D_{11}(x)+C_{12}(x)D_{21}(x))x_1 \\ &\quad +(C_{11}(x)D_{12}(x)+C_{12}(x)D_{22}(x))x_2 \\ y_2 &= (C_{21}(x)D_{11}(x)+C_{22}(x)D_{21}(x))x_1 \\ &\quad +(C_{21}(x)D_{12}(x)+C_{22}(x)D_{22}(x))x_2 \end{aligned} \tag{1}$$

---

†Professor, Associate Fellow AIAA

‡Research Assistant Professor, currently with Department of Mechanical and Aerospace Engineering, West Virginia University, Morgantown, WV, USA

Copyright © 2000 by Mario Innocenti and Giampiero Campa, Published by the American Institute of Aeronautics and Astronautics, Inc., with permission

Matlab and Simulink are trademarks of The Mathworks Inc.

For instance, if we use the two blocks $C(x)$ and $D(x)$ to construct the respective matrices, the resulting Simulink diagram is:

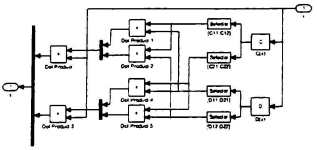


Figure 1. Matrix Multiplication Example

Needless to say, this scheme is not general nor it is easily adaptable if the dimensions involved vary, and especially if they increase above 3 or 4. Where does all this complexity come from?

The main problem in this case is that, although the wires at the output of the blocks $C(x)$ and $D(x)$ carry all the four elements of the two matrices, when we have to multiply the two matrices we have to decompose them into rows or columns. In conclusion, even though constructing systems by connecting blocks has in general several advantages, in some cases (especially when we have to handle nonlinear multivariable systems) it gets very difficult.

Writing S-functions as M-files

Another option is to write directly the S-function that we need using Matlab language. Since Matlab notation is very similar to the standard mathematical notation, this method, when properly used, naturally leads to the most elegant and compact way to describe dynamical systems. For example, in the case of the system considered above, we could write the following M-file:

```
function
[y,x0,st,ts]=mysys(t,x,u,flag)
if flag==0          % initial info
%  0 continue and discrete states
%  2 inputs, 2 outputs
   y=[0,0,2,2,0,0,1];
   x0=[];
   st=[];
   ts =[0 0];     % sample time
elseif flag == 3 % outputs
   C=... {compute C(x)}
   D=... {compute C(x)}
   y=C*D*x;
else
   y=[];
end
```

Without going into the details we will just point out that once we have $C(x)$ and $D(x)$ computed and stored in the variables C and D, it takes just the simple expression

```
y=C*D*x;
```

to compute the output. If we save this file as "mysys.m" then we can use it just by specifying the name "mysys" in the mask of the predefined Simulink S-function block:

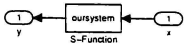


Figure 2. Simulink S-function block

This block will then be equivalent to the scheme in Figure 1. Although we no longer "see" how the system is built directly from the Simulink scheme, when dealing with complex systems this methos seems to be by far the simplest one. Writing S-functions as M-files has only one main drawback: since we use Matlab language, the Matlab interpreter has to be called to evaluate each row of the S-function at every simulation step, this fact prevents the scheme from being used with Real Time Workshop, (which is the standard real time code generator for Matlab) and, last but not least, may slow down the simulation from 10 to 50 times.

There exist compilers that translate M-files to C-files and then to mex (Matlab executable) files, but unfortunately none of them (yet) produce a code that can be used with Simulink, so the only way in which we can use such compilers is the following:

1. write the main S-function as a frame that in turn calls other subfunctions written as M-files themselves.
2. use the compiler on these subfunctions.

This method can help to speed up the the simulation when the subfunctions carry out a significant amount of computation, but since we still need to call the interpreter, we can't use Real Time Workshop yet.

Writing S-functions as C-files

Instead of using Matlab language, we could write the S-function in C language directly, following the "Simulink level 2" conventions [1]. In this way, the S-function could be compiled so that it behaves as a built-in Simulink block, so the interpreter no longer needs to be called, Real Time Workshop can be used, and the simulation is as fast as possible.

Unfortunately, writing an S-function as a C file is in no way as simple as writing it as an M-file, and requires, other than a good C programming language, a careful study on how Simulink works, on how to pass variables and parameters from the Simulink block to the inner S-function and so on. But the biggest problem is that the Matlab language,

and its types are no longer available so we cannot rely on indexing, built in functions, and above all we cannot rely on matrix operations.

In the above example, the only way for us to perform the output computation with a line of code as simply as it was in the case of Matlab, i.e.

```
y = C*D*x;
```

would be to define a class matrix, a subclass vector, and overload the operator *, otherwise we have to write a function mat_product that performs the matrices multiplication, and write something like:

```
mat_product(tmp1,D,x,2,1);
mat_product(y,C,tmp1,2,1);
```

the same is true if we want to translate in C any other Matlab built in operation.

In conclusion, writing the S-function in C it is often the best thing from the performance point of view but unfortunately it's not very practical and requires, some programming knowledge and a considerable amount of work.

Vectorization Blocks

Main Point

The signals that flow through Simulink wires, which in general are the only quantities allowed to vary with time, can be **scalars** or **vectors**.

It is not permitted to carry a matrix through a wire, namely we could certainly carry a p by n matrix as a vector of p*n entries (as we do in Figure 1 with the matrices $C(x)$ and $D(x)$) but all the built in blocks that will process this vector will still treat it just like a vector, so every time we have to process the p by n matrix we have to decompose it into its elements. These decompositions, and the successive compositions make it very uncomfortable to deal with varying matrices in the Simulink environment. All this could be avoided if we had some blocks that would be capable to deal with matrices without decomposing them into elements.

This is exactly the main idea behind the "vectorization blocks". Each input of them is a vector that represents a matrix ordered columnwise according to dimensions provided through its mask. (so row vectors, column vectors and even scalars are just a particular case), and the output is another vector that represents the "output matrix", which is the result of some operations on the input matrices, (product, transposition, norm, pseudo-inversion and so on).

These blocks are based on S-functions written in C (following the Simulink 3 optimized "level 2" conventions) and then compiled so that their execution is as fast as if they were built in blocks. We will present the blocks for matrix transposition and matrix multiplication, since they can cover the implementation of the majority of the control systems. The blocks for pseudo-inversion, inversion, norm, determinant (the inversion block is very useful for simulating non rigid bodies), are currently under development, that is they already exist, but they are not yet operational for use within real time workshop.

Multiplication Block

The first block, which is based on the S-function "vmult.c", performs matrix multiplication. This block accepts as inputs two vectors each one representing a matrix whose elements are taken columnwise, and accepts as parameters 3 numbers:

1. n = number of rows of the first matrix.
2. p = number of columns of the first matrix = number of rows of the second one.
3. m = number of columns of the second matrix.

The two matrices are then reconstructed from the two vectors, and the first one is (left) multiplied by the second one, the elements of the n by m resulting matrix are then reordered columnwise into the output vector. The following picture shows the appearance of the block:

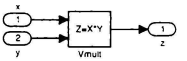


Figure 3. Multiplication Block

By double clicking on the block the following mask appears:

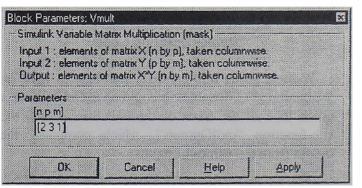


Figure 4. Multiplication Block Mask

The vector [n p m] contains the dimensions that the function needs to build the matrices from the inputs.

Transposition Block

The second block, which is based on the S-function "vtrsp.c", performs matrix transposition. This block accepts as input one vector representing a

matrix whose elements are taken columnwise, and accepts as parameters 2 numbers:

1. n = number of rows of the matrix.
2. m = number of columns of the matrix.

The matrix is then reconstructed from the two vectors, and then the elements of its transpose are reordered columnwise into the output vector. The following picture shows the appearance of the block:

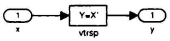


Figure 5. Transposition Block

By double clicking on the block the following mask appears:

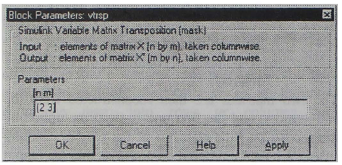


Figure 6. Transposition Block Mask

The vector [n m] contains the dimensions that the function needs to build the matrix from the input.

Redundancy

Knowing the number of inputs, the number of dimensions to be typed in the masks of both blocks has one redundancy. Retaining the present level of redundancy was justified because it allowsdouble-checking and therefore helps preventing errors in connecting the blocks to one another.

Implementation Code

Without going into the details on how to write an S-function in C, we think it could be useful to have a quick look at the heart of the two S-functions: the sub-function "mdlOutputs", which computes the output from the values of inputs.

Vmult.c

Here is the code of mdlOutput for the S-function vmult.c:

```
static void
  mdlOutputs(SimStruct   *S,   int_T
tid)
{
int_T   i, j, k, *n
  =ssGetIWork(S);
real_T *y
  = ssGetOutputPortRealSignal(S,0);
InputRealPtrsType up1
  =
ssGetInputPortRealSignalPtrs(S,0);
InputRealPtrsType up2
  =
ssGetInputPortRealSignalPtrs(S,1);
for(i = 0; i < n[0]; i++)
   for(j = 0; j < n[2]; j++)

for(y[i+j*n[0]]=0,k=0;k<n[1];k++)

y[i+j*n[0]]+=(*up1[i+k*n[0]])*

(*up2[k+j*n[1]]);
}
```

A quick explanation of the code is necessary. Integers i, j, k, are indexes, n is a pointer at the three elements vector of the dimensions, y is a pointer at the output vector, up1 and up2 are pointers at the input vectors. The three for loops perform a matrix multiplication, in particular the first two loops select the row and column of each element of the output matrix, and the inner loop performs the scalar product between the relative row of the left matrix and the relative column of the right matrix.

It is important to notice that we are deliberately using a fast indexing that allows us to handle matrices directly in their vectorized form without actually losing computation time on array construction. Thus the code is very optimized for speed.

Vtrsp.c

Here is the code of mdlOutput for the S-function vtrsp.c:

```
static void
  mdlOutputs(SimStruct   *S,   int_T
tid)
{
int_T  i, j, k, *n
  = ssGetIWork(S);
real_T  *y
  = ssGetOutputPortRealSignal(S,0);
InputRealPtrsType  up
  =
ssGetInputPortRealSignalPtrs(S,0);
for(i=0;i<n[0];i++)
   for(j=0;j<n[1];j++)
       y[j+i*n[1]]=(*up[i+j*n[0]]);
}
```

Integers i, j, k, are indexes, n is a pointer at the two elements vector of dimensions; y is a pointer at

the output vector, up points to the input vector. The two for loops perform the matrix transposition just by exchanging indexes. As in the preceding case, matrices are handled directly in their vectorized form.

Examples and Considerations

Two multiplications

Let us reconsider the case in which we have the output depending from the state vector in this way:

$$y = C(x)*D(x)*x$$

we can construct this function using the Varying Matrices Multiplication Block in this simple way:

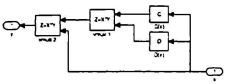


Figure 7. Two matrix Multiplication with vmult

A quick comparison between Figure 1 and Figure 7 is sufficient to appreciate the simplification; moreover, this new Simulink scheme is completely general, that is, if we change the dimensions of the matrices involved, we only have to write the new dimensions into the two Varying Matrices Multiplication Blocks.

MIMO Adaptive Sigmoidal Neural Network

The following example is more complex, here we used the two blocks to implement a continuous time adaptive multilayer MIMO sigmoidal neural network [2], [3], [4] which is represented by the following equation:

$$\begin{bmatrix} \dot{N} \\ \dot{M} \\ v_{ad} \\ \dot{\bar{v}} \end{bmatrix} = \begin{bmatrix} -G[\bar{x}\zeta M^T\sigma'(N^T\bar{x}) + \lambda|\zeta|N] \\ -F[(\sigma(N^T\bar{x}) - \sigma'(N^T\bar{x})N^T\bar{x})\zeta + \lambda|\zeta|M] \\ M^T\sigma(N^T\bar{x}) \\ K_v(\|Z\|_F + \bar{Z} + K_z)\zeta \end{bmatrix}$$

There are two inputs to the net, the first is x, which should contain information necessary to the net to approximate a function $\Delta(x)$, and the second is ζ, which is the "error" term that actually drives the network.

The network output is composed of two vectors: the adaptive term v_{ad} and the robustifying term $\bar{v}$, which is made proportional to ζ. The network states are exactly the interconnection weights between different layers, namely all the entries in the two synaptic matrices M (connecting hidden layer to output layer) and N (connecting input layer to hidden layer). The output of the hidden layer is the vector

$$\sigma(N^T\bar{x}) = [1 \quad \sigma_1(n_1\bar{x}_1) \qquad \sigma_h(n_h\bar{x}_h)]^T \qquad (2)$$

where $n_1 \ldots n_h$ are the rows of N^T and σ is the activation function, which is chosen to be:

$$\sigma_i(z) = \frac{1}{1+e^{-az}} \qquad (3)$$

The matrix $\sigma'(N^Tx)$ is the Jacobian of $\sigma(N^Tx)$ with respect to N^Tx evaluated at N^Tx. Z is defined as diag(M,N), and $|Z|_F$ is its Frobenius norm. All other values are constants. Specifically F and G are the learning rates of the first and second layers, and λ is commonly known as the e-modification gain. Constants required for the robustifying term are K_v and K_z. Finally, $\bar{Z}$ is an upper bound for $|Z|_F$.
The following picture, Figure 8, shows the realization of the net as a Simulink scheme.

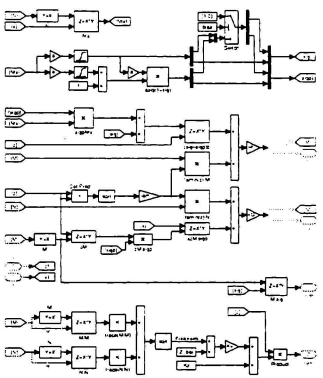


Figure 8. Neural Network Scheme

The upper part of the scheme is devoted to the computation of the hidden layer output (4) and its jacobian matrix. The central part is the heart of the function in that it computes the derivatives of the states N and M, and then integrates them. Finally the lower part of the scheme contains the blocks for the output computation, which involves also the computation of the Frobenius norm of Z.

This scheme has been successfully used to simulate in real time several neural network based

controls [2], [3], although slightly faster, its version as a C program was 420 lines long and had several limitations, a full-general and optimized version would probably be much longer.

Rotation Matrices Library

The multiplication and transposition blocks were used to build a library of rotation matrices, based both on Euler angles and quaternions. These matrices are a classical example of very required varying matrices blocks, since they are very useful for the simulation of rigid bodies in a full 3D environment.
Since their size is fixed, and the elements are just 9, they can be easily implemented without the need of the blocks presented in the paper, but the use of these blocks can make their implementation extremely fast and straightforward.
The picture below shows the rotation matrix that transforms body fixed coordinates into earth fixed coordinates, given the orientation of the body with respect to earth in roll, pitch and yaw angles [5].

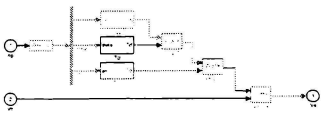


Figure 10. Body2Earth Rotation Matrix

The scheme above is clearly related to the formula:

$$ {}^{e}R_{b} = e^{\psi S\left(\begin{bmatrix}0\\0\\1\end{bmatrix}\right)} e^{\vartheta S\left(\begin{bmatrix}0\\1\\0\end{bmatrix}\right)} e^{\phi S\left(\begin{bmatrix}1\\0\\0\end{bmatrix}\right)} $$

The Multiplication, Transposition, and Inversion blocks, are currently being used by the first author in a recursive frequency domain identification scheme to be used for on-line flight parameter estimation and self adaptive flight control.

Modularity issues

In our opinion, there is also another fact, less evident but very important, that makes a totally Simulink-based implementation best than one which is partially coded in Matlab or C.
Often, the person who designs a Simulink scheme, has barely taken a quick computer programming course in his/her studies, so it's pointless to expect that he/she can write efficient code with good modularity and reusability characteristics.
In these cases, a totally Simulink-based project generally retains better modularity and reusability characteristics than a partially coded project.
Since as project complexity grows modularity becomes critical, this fact alone can, in our opinion, become the most compelling reason to adopt a totally Simulink-based implementation.

Where to find the blocks

The four blocks, along with all the examples in the paper and instructions files for their use, can be freely downloaded at the official mathworks ftp site: ftp://ftp.mathworks.com/pub/contrib/v5/simulink/matrix_library

DSP Blockset

If the DSP toolbox is available, then the DSP matrix library can be used, which is very efficient and complete. The presented blocks are fully compatible with the ones of the DSP matrix library, and of course all the above considerations on the opportunity of using the presented blocks, applies as well to the blocks in the DSP matrix library.

Conclusions

In this paper, we briefly explained Simulink's S-functions mechanism, which is used to represent and simulate dynamical systems, in particular we have shown the three common methods to build S-function, and underlined that in some cases neither of them can be at the same time easy and suitable for real time simulations.
Two Simulink blocks were proposed that solve this problem by allowing varying matrices to be handled directly in Simulink environment. Their aspect, their use, and the most important part of their code were explained, and some examples were given.
Finally reasons that render the use of these blocks highly desirable were given.

References

[1] Matlab 5.3 and Simulink 3 manuals, The Mathworks Inc.

[2] G. Campa, M. Sharma, A.J. Calise, M. Innocenti, "Neural Network Augmentation of Linear Controllers with Application to Underwater Vehicles" *American Control Conference 2000*, June 2-4, 2000 San Diego.

[3] F. Nardi, R.T. Rysdyk, A.J. Calise, "Neural Network Based Adaptive Control of a Thrust Vectored Ducted Fan," *AIAA Guidance, Navigation and Control Conference*, AIAA 99-3996, Portland, OR, 1999.

[4] F.L. Lewis, S. Jagannathan, A. Yesilderek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*, Taylor & Francis, London, 1999, pp. 150-170.

[5] Guidance and Control of Ocean Vehicles, Thor J. Fossen John Wiley and Sons 1994.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37270

AIAA-2000-4396

USING HIGH LEVEL ARCHITECTURE (HLA) FOR DISTRIBUTED SIMULATION OF STOCHASTIC PETRI NETS*

Kpatcha M. Bayarou and Helena Szczerbicka
E-Mail: {kpatcha; helena}@informatik.uni-bremen.de
FB 3: Department of Mathematics and Computer Science
University of Bremen, P.O.Box 330440, 28334 Bremen, Germany

Abstract:

We present the use of HLA for distributed simulation of stochastic Petri Nets (SPN). We decompose a SPN model into submodels and assign them to Logical Processes (LPs). LPs are surrogates for physical processes of the modeled "physical system". LPs are distributed on heterogeneous nodes of a computer network and communicate with each other by passing messages through HLA-RTI (run time infrastructure). Using HLA paradigm we assign LPs to federates and the whole simulation SPN model to a federation. We evaluate some events synchronization techniques, time management schemes and strategies provided by HLA regarding their impact on the simulation speed.

Keywords: (S)PN, HLA, Time management, Distributed Simulation

Introduction:

Petri Nets (PN) are a graphical and mathematical modeling technique for formal description of systems whose dynamical behavior have properties like concurrency, synchronization, conflicts and mutual exclusion, which are seen as typical features of distributed environments. In this paper we use a type of Petri Nets called stochastic Petri Nets (SPN), which are derived from time augmented Petri Nets by assigning negative exponential (memoryless) randomly distributed firing time to the transitions. SPN models are characterized by the fact that their qualitative behavior is identical to that of their underlying PN model without any temporal specifications, so that the reachability and structural analysis results obtained from the PN model are valid also for the SPN.

In recent times the potential benefits of distributed modeling and simulation have been widely recognized in industry, in academic research and other application fields. The main difficulties in the area of distributed simulation are how to manage events without causality violation and how to achieve the communication between the distributed components which compose the simulation model. To cope with these difficulties among others, the HLA concept was developed by the Defense Modeling and Simulation Office (DMSO) of the Department of Defense (DoD). The main aim of HLA is to support interoperability between simulations and reusability of components. HLA is a component integration standard for distributed simulation and becomes also an IEEE standard No. P1516 by the Simulation Interoperability Standards Organization (SISO). Originated from the Military domain, HLA however also addresses the needs of the civil simulation community whenever the interoperability and reusability for distributed and cooperative software is demanded.

In this paper we present the use of HLA for stochastic Petri Nets models. We decompose a stochastic Petri Net model into submodels and assign them to LPs. LPs are surrogates for the parts (physical processes) of the "physical system" the submodels describe. Hence each logical process contains a portion of the state corresponding to the physical process it models, as well as a local clock that denotes how far the process has progressed. The LPs are distributed on heterogeneously connected nodes of a computer network and communicate with each other by passing messages. For the intercommunication between the LPs we use the RTI

* Copyright© 2000 by Bayarou.
Published by The American Institute of Aeronautics and Astronautics, Inc. with permission

(run time infrastructure) provided by HLA. To conform with the HLA paradigm we assign the LPs to the federates (these are HLA compliant simulation programs) and the whole simulation SPN model to a federation.

The central goal of parallel and distributed simulation is and remains the speed-up gain in comparison to the sequential simulation. To achieve this goal, researchers have invested efforts in developing and implementing different communication protocols. With the HLA, an alternative software architecture is provided which is used to investigate possible speed-up gain for distributed execution of a Petri Nets model exhibiting inherent parallelism.

Although speed-up is not a primary aim of HLA, Performance in the use of HLA for distributed simulation is an important issue. In this paper we study the problem of accelerating the sequential execution of SPN model by distributing its submodels on multiple heterogeneous processors. We evaluate different events synchronization techniques, time management schemes and strategies provided by the HLA regarding their impact on the simulation speed.

We have implemented this simulation in C++ based on UNIX-Xwindows environment running on Solaris 2.7 Sparc and the RTI version RTI1.3-NG. As an example we simulate a readers and writer problem.

1. Petri-Nets

1.1 Basic Definitions

Definition 1.1
A Petri net its a 5-tuple, $PN=(P,T,F,W,M0)$ where:
$P=\{p1,p2,...,pn\}$ is a finite and non-empty set of places,
$T=\{t1,t2,...,tm\}$ is a finite and non-empty set of transitions
$F\subseteq(P\times T)\cup(T\times P)$ is a set of arcs
$W: F \rightarrow IN$ is a weight function,
IN means the set of natural number
$M0: P \rightarrow IN0$ is the initial marking,
$IN0$ means the set of natural number including {0}
$P\cap T=\varnothing$ and $P\cup T\neq\varnothing$

Since its inception Petri Nets have been extended to capture many important aspects of modeling large-scale systems. An introductory survey of a variety of Petri Nets is available[14] and there are tools dealing with some of the existing Petri Nets classes[2,3]. For the graphical representation of a Petri Nets, circles are used for places, rectangles or simple bars are used for transitions, arcs are used for the elements of the set F and dots or natural number for tokens (an item indicating that something lies in a place). Places are used to model e.g. conditions, data (input or output), resources, buffers etc.. Transitions are used for events, signals, task or job, processor activity etc... Arcs indicate the relationship between places and their corresponding transitions. The values of the weight function quantify/qualify these relations whereas the values of the marking function determine the dynamics of the modeling systems.

According to the definition of the set F a transition may have places as its inputs or outputs and a place may also have transitions as inputs or outputs. The dynamics are based on locality principle that means a transition activity affects only its local environment. If a transition is active, then this activity consists of removing tokens from its input places and creating tokens in its output places. Due to the described dynamic behavior, the marking can be seen as a State of the modeling system, with values of $M0$ representing the initial State. We can present $M0$ as a vector whose dimension is the cardinality of the set of places and the value of $M0(pi)$ as the entry of its *ith*-component. Generally the state vector is written as $M=(M(p1),...,M(pn))$ for any state M of the net. If a transition is enabled, it fires and it deletes tokens from its input places and creates tokens in its output places. A transition is enabled if its input places contain more tokens than the weight of the arcs linking them to this transition. Further more the transition must be able to deposit the necessary number of tokens indicated by the weights of the arcs linking it to its output places whiles respecting the capacities of those places. A Capacity of a place is a number $C(p)$ which indicates the maximal number of tokens this place may contain. If the capacity is not specified, the place can contain an infinite number of tokens. The firing of a transition leads to a new marking of the net which may also be its old or initial state. Using enabling and firing rules one can play a so called Token game to simulate the behavior of the modeled system, depicting possible reachable markings (states) M starting from the initial marking $M0$. The whole set of reachable markings is referred to as Reachability set, denoted by $RS(M0)$ which represents the set of states of the modeled system. The reachability set contains no information about the transition sequences fired to reach each marking. This information is contained in the reachability graph $RG(M0)$[8], where each node represents a reachable state, and there is an arc from

M_i to M_j if the marking M_j is directly reachable from M_i. The expression $M_i[t > M_j$, means t is enabled in M_i and its firing leads to M_j. The reachability set is the basis for reachability graph $RG(M_0)$[10], where each node represents a reachable state, and there is an arc from M_i to M_j if the marking M_j is directly reachable from M_i. If $M_i[t > M_j$, the arc is labeled with t.

1.2 Stochastic Petri Nets

In the performance or quantitative analysis of time behavior of a system, the notion of time in Petri Nets is necessary[14, 4]. In this paper we consider stochastic Petri Nets with exponential distributed firing time delays for transitions i.e. only transitions are associated with time parameters. Once a transition is enabled, it will fire after a certain (not fixed) time interval has elapsed. If two transitions are in structural conflict, the firing sequence is based on "race-policy". For the race models, tokens are not reserved by any transition. Once a transition is enabled, it waits for its firing time to elapse and then fires immediately assuming it is still enabled at that time. With this strategy the faster a transition the more likely it might disable other transitions leading to enabled transitions competing for tokens. It is also said that they "race" for tokens.

As a definition of the stochastic Petri Net (SPN) (we are concerned with continuous time evolution of the stochastic variable) we add a set $\Lambda = (\lambda 1, \cdots, \lambda m)$ to the definition 1.1 of Petri Net to obtain $SPN = (PN, \Lambda)$ The firing time of the transition t_i is exponentially distributed with distribution parameter λ_i. Stochastic Petri Nets with exponentially distributed firing delays models have the characteristics that their qualitative behavior is identical to that of their underlying PN with no time specifications, so that the reachability and structural analysis results obtained from the PN model are valid also for SPN model[10]. Moreover SPN models are isomorphic to continuous-time Markov chains; therefore the performance or quantitative analysis of SPN can be carried out in a straightforward manner by analyzing the corresponding underlying Markov process[4].

1. 3 Spatial partitioning SPN:

One way of accelerating the simulation runs is the use of computational resources particularly multiple processors operating in parallel and/or distributed on different nodes of processors. In this paper we consider the event-driven simulation i.e. the observation of the simulated system will be done at event occurrence instants. To make use of the parallelism available in the physical system being modeled, the simulation model has to be decomposed into model components (submodels) called logical processes (LP), such that the decomposition directly reflects the inherent model parallelism[1]. The critical problem posed by the parallel and distributed simulation is how to ensure the preservation of causal order. It is required that every LP adhere to processing events in non-decreasing event time (timestamp) order only; this requirement is referred to as local causality constraint (lcc)[11, 7]. There are two main categories of mechanisms for dealing with the lcc: conservative methods strictly avoid lcc violations and optimistic methods hazardously process events.

Using stochastic Petri Nets Model for distributed simulation, we build a set of submodels from the sequential simulated stochastic Petri Nets model and assign them to logical processes LPs which in turn will be assigned to federates. For the spatial partitioning of the stochastic Petri Nets model we use method[6]. The main potential contribution of the use of Petri Nets model description for distributed discrete event simulation is a capability to define causality relations explicitly through the graph structure, thus allowing one to relax constraints on timestamp ordering for the processing of events that are not causally related. Based on this consideration we can expect to be able to increase the parallelism of a distributed discrete event simulation with respect to the standard approaches by simply taking the structure of the Petri Nets model into account. In this paper every LP applies one and the same discrete event strategy (conservative) to the simulation of the local subnet. The subnets are from type Petri Net, the same as the Petri net to be partitioned. Therefore a subnet has the form: $SPNk = (PNk, \Lambda k)$ which indicates the logical process k of the stochastic Petri Nets model $SPN = (PN, \Lambda)$. The set of places and the set of transitions of two different logical processes must be disjoint.

The advantage of preserving the Petri Net type for the submodel is the application of one and the same simulation engine (program) for all submodels. In our consideration the simulation engine can also be used for the Petri net model to be decomposed. Apparently the decomposition of a SPN model has a strong impact on the distributed discrete event simulation performance. Small sized submodels raise high communication demands whereas large scale submodels can help to clamp local SPN behavior inside one LP. The partitioning[6] is based on the output arcs of transition, thus on the set $(T \times P)$ of

the arcs. The reason is to avoid distributed conflict resolution, since in this way conflict resolution always occurs internally to an LP, therefore involving no communication overhead. The generality of the partition is limited in such a way that conflicting transitions together with all their input places are always put together in the same LP, in order to avoid distributed conflict resolution.

Minimum region[6] characterizes the smallest unit of subnet which can be obtained through the partitioning of the SPN. This definition makes use of the conflict set CS(t) of a transition. As conflict set of a transition t, we mean the set of transitions which share input places with the transition t. This conflict set is also called structural conflict set.

Definition 1.2 Minimum region (subnet):

A minimum region $MR_k=(P_k,T_k,F_k,W,M_0)$ of some $PN=(P,T,F,W,M_0)$, where :

$$T_k=\begin{cases}\{t_i\}, \text{ if } t_i\in T \text{ is a single transition i.e.} \\ \qquad\qquad \text{not participating to a conflict} \\ CS(t_i), \text{ whole conflict set iff } t_i \\ \qquad\qquad \text{participates in some conflict set}\end{cases}$$

P_k is the set of input places for all transitions in T_k

and

$F_k \subseteq (P_k \times T_k)$

As communication channels, arcs which are not embedded in the minimum regions are used.

In our concept we waived the restriction that the source of a channel has to be a transition and the sink a place. This restriction[6] is made with respect to a given communication protocol.

Our reason for not upholding the restriction is because a decomposition may lead to situations where a place may be a source of a communication channel. We consider a LP as being composed of places and transitions without empty arcs allowing a LP to have places and transitions to be on the output border (see figure 4). A LP may also consists of minimum regions. We strictly avoid conflict propagation i.e. two transitions which share at least one place as their input place belong to the same LP. From an output place there is only one message that a LP can send to an adjacent LP at a given time as shown in the figures 2.a-2.c.

2 HLA

The HLA[15] is a software architecture for creating computer models or simulations out of component models or simulations. The combined simulation system from the constituents is named a federation. Each constituent is referred to as a federate. A session of a federation executing together with its components is called federation execution. The main components of a federation are: a software controlling events flow and time management named RTI (Runtime Infrastructure), a common object model for data exchange between federates involved in a federation called FOM (Federation Object Model) and a number of federates. The federates and RTI are software while FOM is data created by the federation developer. From the perspective of the HLA a federate is defined by its single point of attachment regardless of how the federate is conceived to run (sequential, parallel or distributed) and regardless of what and how many entities it models. As a software architecture HLA is a standard and not a particular implementation, therefore it is defined by a set of documents. HLA standard[5] has three main parts:

HLA rules, Object model template (OMT) and Interface specification.

The HLA rules are principles and conventions that must be followed to achieve proper interaction of federates during a federation execution. These rules are design principles for the interface specification and the object model template (OMT) and describe the responsibilities of federates and the federation designers.

The HLA object model template (OMT) prescribes the allowed structure of the federation object model (FOM), which every federation must have i.e. OMT is a meta-model for all FOMs.

The interface specification is the specification of the interface between federates and the runtime infrastructure (RTI). The interface between the RTI and federates is standardized but the implementation of the RTI could take a variety of forms.

HLA is already adopted by international organizations as a standard e.g. OMG (Object Management Group) under "Facility for Distributed Simulation Systems" and IEEE (Institute of Electrical and Electronics Engineers) HLA IEEE 1516.

In order to facilitate the adoption of the HLA DMSO (Defense Modeling & Simulation) has defined a development process for federations called FEDEP (Federation Development and Execution Process) which involves five basic steps: concept development, federation design, federation execution implementation, testing and operations. In order to cope with causality problem HLA provides conservative and optimistic mechanism in the form of time management services.

3 Simulation

For the purposes of simulation of stochastic Petri Nets we have developed a HLA-Based simulation engine, in which the Federation is constituted from stochastic Petri Net submodels that are obtained by

partitioning a stochastic Petri Net model. In our concept, we consider two levels of system description: (1) the real world domain is described in terms of a simulation model consisting of subnets. For this purpose we use OOAD (object-oriented analysis and design) concepts. (2) HLA object models, in which the system description focuses specifically on requirements and capabilities for the exchange of information from federate to federate. For this purpose we define (see figure 1) two interaction classes: communication interaction class and update interaction class.

In our Federation Object Model (FOM) the description of all shared information which are essential to a particular federation such as objects and interactions is made according to the Object Model Template (OMT) definition. For communication purposes (the messages the federates exchange via RTI), we use Token-messages. A Token-message has a dedicated coded form, *<Id, destination, #token, sourceLP>* in which the following information is contained:

(1) an *Id* denoting the type of token message i.e. a flag which indicates if the token is to be deposited in a place or a token for enabling a transition located in the destination federate, because we have not excluded the situation where a token may contribute to the enabling of a transition in a destination federate. (2) *destination* indicating a place or a transition. In the implemented concept, the places and transitions maintain their labeling as in the original undecomposed SPN model. (3) A marker *#token* that indicates the number of tokens to be deposited in a place in the adjacent federate, or number of tokens according to the arc function that the destined transition needs to be enabled. (4) Source *sourceLP* indicating the generator of the token message.

An interaction is sent at one time through the RTI to other federates. It may represent an occurrence or event in the simulation model of interest to more than one federate. An interaction may be defined to occur at a point in simulated time. A federate sends an interaction; other federates receive the interaction. The interaction has no continued existence after it has been received. Each interaction carries with it a set of named data called parameters[5]. Accordingly, we defined the Token-messages may occur in simulation time. We define for our purpose two kinds of interaction classes. One class containing the token messages which is used for communication between the federates via RTI producing messages of TSO (time stamp order) category and another class of interaction which affects only places in the system and producing message of the RO (receive order) category. The messages of the latter category are used to update places in the system when needed.

Both interaction classes: ComInterClass and UpdateInterClass are embedded in the hierarchical structure organized in FOM with a single root called InteractionRoot (defines no default parameter). Therefore they are derived from InteractionRoot and manager class as shown in the figure 1

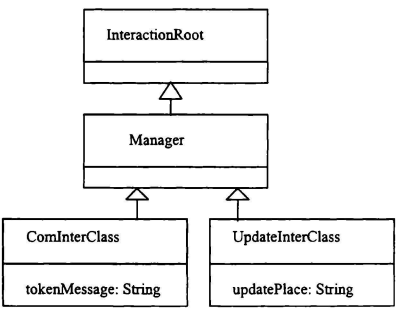


Figure 1 Interaction classes in UML notation

Situations conducive to the use of RO message

As stated above the interaction class UpdateInterClass produces interactions serving for updating the places when needed. In this section we precise the situation when these interactions are necessitated. To explain this we consider two logical processes LP1 and LP2 (figure 2.a).

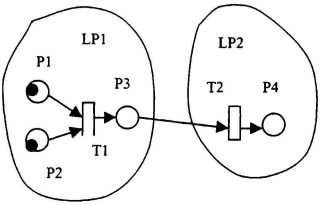


Figure 2.a Situation for using RO messages

In the situation as shown in this figure, since the transition T1 is enabled if its delay time elapses it will fire and removes tokens from P1 and P2 and creates a token in P3. After doing this LP1 sends a token message to LP2 advertising to it that there is a token in P3 destined for T2, LP2 notices the message and checks the enabling rules for T2. Since in this case T2 is enabled it will fire when its delay time elapses. T2 fires but is not allowed to remove a token from P3. T2 puts one token into P4, the token from P3 however must be removed as quickly as possible. For this purpose, LP2 sends RO message to LP1 mandating him to remove the token (which according to firing rules should be consumed when the transition fires) from P3. After doing this we get the regular situation as shown in figure 2.b

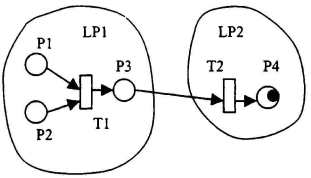


Figure 2.b: Right Situation after using RO messages

If this situation were not considered, we would get a marking which is not included in a reachability set of a sequential Petri Net model as shown in the figure 2.c.

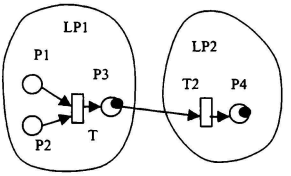


Figure 2.c Bad situation necessitating RO messages

3.1 Synchronization of the PnFederation

We need to synchronize the firings of transitions in spatially different net parts to avoid timing inconsistencies i.e. concurrent Petri Net execution must ensure correctness in the sense that the partial ordering of transitions firings produced is consistent with the total event ordering that would be used by a sequential execution. Therefore the PnFedrates in a PnFederation must involve time management which is provided by the HLA. The HLA time management can only ensure that events are delivered to the federates in the correct order (maintaining firings timings consistency). In PnFederation each PnFederate is both time-regulating and time-constrained. A time-regulating federate is one whose advance of logical time regulates the rest of the federation (specifically those federates that are time-constrained). A time-constrained federate is one whose advance of logical time is constrained by the rest of the federation (specifically those federates that are time-regulating)[5]

We describe the way we use Time Management Services to manage each PnFederate (SPN-Fed). HLA provides time management services which operates event-driven, time stepped, parallel discrete event simulation or wallclock time driven[12]. In an event driven simulation the federate processes local events and those generated by other federates in time stamp order i.e. depending on the time when the event was generated. For time stepped simulation, each time advance made by the federate is of some fixed duration Δt of simulation time (time step). It means that the simulator (federate) does not advance to the next time step until all simulation activities associated with the current time step Δt have been completed. Using parallel discrete event simulation paradigm, federates executing on multiprocessor systems must be synchronized internally using either a conservative or optimistic synchronization protocol[13].

The mechanism upon which the access to these Time Management Services is done, reposes on the

"request-grant"-principle i.e. a federate requests services and RTI grants when possible.

At granted time T, each PnFederate holds a future event list that is organized in increasing event time (timestamp) order. In our implementation we distinguish between two endogenous (internal) events consisting of (1) the scheduled firing of a transition and (2) the firing time (time stamp).
The following steps are performed in the PnFederation according to the HLA time management services:
1. PnFederate invokes the time management service called Next-Event-Request to request its logical time to advance (to the first event time)
2. RTI delivers some number (possibly zero) of messages to the PnFederate. The RTI invokes the PnFederate procedure called Receive Interaction to deliver interaction events which are as stated above Token-messages. We have implemented Receive Interaction procedure in the PnFederate Ambassador.
3. RTI invokes the primitive called Time Advance Grant provided by HLA to indicate to the PnFederate that its logical time has been advanced.

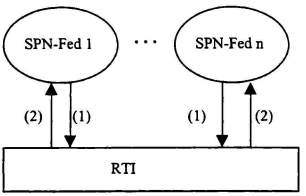


Figure 3: Schema of Time Mangement Services Invocation

In this schema (figure 3) the arrows labeled (1) show the Next Event Request invocation which need a time parameter T e.g. NextEventRequest(T) from the PnFederate, and those labeled (2) show the invocation of the service Time Advance Grant which RTI invokes.
Two states are described by federates involved in time management: time Granted state and time Advancing state. Thus we distinguish between time Advanced phase and time Granted phase.

- Time Advanced Phase
  In this section we describe, how a PnFederate performs its simulation activities. A typical scenario which may occur is that of the empty list i.e. a PnFederate may have an empty event list either at the beginning of the simulation or after the PnFederate has carried out some of its simulation activities. This scenario of empty event list may be due to the fact that a PnFederate does not have an enabled transition (at the considered time) or after a few simulation runs all transitions in the PnFederate become disabled. In such situations, the PnFederate is in the Time Granted state (federate receives no TSO events in the Time Granted state). According to the HLA concept, if a federate expects to receive TSO events from other federates in order to be able to carry out its simulation activities, it must be in the Advancing state. Therefore from Time Granted state to Time Advancing state, a request to advance its logical time must be made by the PnFederate. But a time parameter T must be passed to the RTI procedure NextEventRequest; in the case where the event list is empty which time parameter T should be passed to the time management service NextEventRequest? To cope with this problem we compute the PnFederate's LBTS and add the smallest delay of the transitions located in PnFederate i.e. $T = LBTS + \tau$, where τ is the smallest next delay of an input transition in the LP simulated by the PnFederate. If the event list is not empty there is no problem when using NextEventRequest, since the time stamp of the event with smallest time stamp is passed as time parameter.
  If the event list is not empty and the transition on the output border (see figure 4) fires the PnFederate will, after firing, check whether the Look-ahead value has changed and if this is the case, it is actualized.

- Time Granted Phase

  RTI indicates that the PnFederate's logical time has been advanced. At this time two points of view are to be taken into account:
  If the granted time T' is smaller than the requested time T i.e. $T' < T$, then we have token messages (TSO messages) with time stamp T' from another PnFederate. So we process these messages at the granted time T'. After processing these events we schedule new internal events, if new transitions are enabled, and then request to advance the PnFederate's logical time again. We process the received RO messages (if there are some), and update the PnFederate marking vector.
  IF the granted time T' is equal to the requested time T i.e. $T' = T$, we fire the transitions associated with this time T, and check whether the Look-ahead value has changed and if this is

the case, actualize it. The Look-ahead problem will be discussed in the next section.

3.2 Look-ahead Computation

Definition of Look-ahead:
If a logical process at simulation time T can only schedule new events for adjacent LPs with time stamp of at least $T + L$, then L is referred to as the Look-ahead for the logical process[13]. Look-ahead is very important for simulation requiring guaranteed message ordering services to achieve acceptable performance. In the HLA, a single Look-ahead value L is declared by each federate. This value may change at runtime, but reductions in Look-ahead do not take effect immediately[12].

We describe how we exploit this definition of Look-ahead in our PnFederation. Two kind of outgoing (exogenous) messages are defined as stated above: a TSO Token-message destined for a place or transition and a RO message destined for a place as a mandate for a PnFederate to remove a token from its place (see above). Additional to these two messages there are internal (endogenous) messages which are firings of transitions assumed they are enabled. If a scheduled transition fires at its scheduled time T it can only that the RTI has granted this time T. Petri Nets have properties based on locality principle. For a PnFederate if a transition fires and it has some successor transitions which are output border transitions, then the Look-ahead is computed as the minimum delay firing time of them i.e. $L = \min(\pi)$ where π is the firing delay of any transition t_i belonging to the output border (see figure 4).

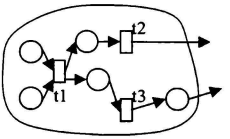


Figure 4 Explaining Output border transitions

In this (figure 4) t2 and t3 are both output border transitions, thus if t1 fires the Look-ahead is computed as explained above. If one of the transitions on the output border fires the PnFederate will check after firing whether the Look-ahead value has changed and if this is the case, it is actualized.

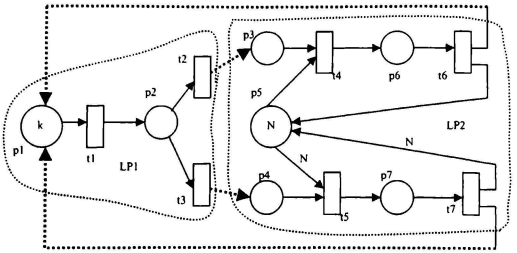


Figure 5 Simulated stochastic Petri Nets model with partions LP1 and LP2

5. Application Example

We use the Reader/Writer simulation model for testing the HLA-based simulation engine.

5.1. Application model

In the figure 5 the stochastic Petri Net model describes the Readers and Writers system. The model comprises seven transitions with exponential firing time distribution with parameter λi. In this model k number of processes depicted in p1 want to access to N resources depicted in p5. After firing t1 which models the entering action, a process enters the place p2 where it makes a choice if its access to p5 is reading or writing. Therefore reader entering p3 (readers waiting queue) waits until the firing time of t4 elapses and starts reading. The transition t4 models the start reading process. The place p6 models reading and the transition t6 the end of reading. If t6 fires it releases the occupied resource and generates another process. If a process enters p4 which models the writers queue then there are two possibilities: (1) If there is any reader waiting in p3 and t4 starts firing the writer must wait. (2) If the transition t5 fires it removes all N resources from the p5 and after it finishes writing (p7 models the writing state and t7 the end of writing) it releases N resources back to p5 and generates another process. The number of processes k and the number of resources N are constant in this system when chosen. That means k and N must be given at the beginning of the simulation. In the following section we experiment with the HLA-based simulation engine of the above model and discuss results by setting various parameters.

5.2. Simulation Results

In this sections we describe the situations observed while simulating and the obtained results.

- System environment: We use SUN SPARC Model SUN Ultra 60 with SunOS 5.7 as operating system.
- Implementation: For the implementation of the HLA-Based simulation engine we use C++ and the RTI version RTI-1.3-NG provided by SDC (software distribution center) of DMSO.

We run the simulation on three nodes of computers. The RTI has been run on a server and the Petri Nets federates on two other computers.

We have performed various experiments changing parameters k, N, λi of the model. We ran the simulation at various times (1) in the early morning and in the night, when there were not many users on the network environment and (2) in the core time when there were many users. In all these situations we have observed that the distributed simulation was slightly slower than the sequential simulation in a ratio less than 50%.

There were violations of causality constraints for certain parameters of time firing.

We coped with this situation by adding a negligible (regarding to the firing time) amount of the time ε to the time of the Token-message while sending this Token-message. This ε is subtracted when the receiver receives the message to keep the scheduling time correct. This resolution has no negative effect on the local time of each federate. Then the simulation may schedule Token-messages ε sec into the future, providing a Look-ahead of this amount. Thus leading to a "Tolerance to temporal inaccuracies"[12].

After following the procedure as described above the simulation has been performed without any causality

violation but taking more run-time because of a too small look-ahead. We expect better results with optimistic simulation time management mechanisms. This is left as further investigations in our research.
We suppose that the observed causality violation depends on the stochastic character of the model which influences the instantaneous look-ahead computation. According to HLA the reductions in look-ahead do not take effect immediately.

6. Conclusions and further Work

In this paper we present an experience of HLA-based distributed simulation of stochastic Petri Nets models. Stochastic Petri Nets model is decomposed and the parts obtained are assigned to logical processes which are used as federates in a HLA-based simulation engine. We have concentrated our investigation on how to use the HLA capabilities for stochastic Petri Nets simulation. Two aspects are crucial in this work (1) the description of data exchanging among the federates to carry out the simulation and (2) time management for keeping the causality order in the simulation. Concerning (1) we describe how we use the interaction classes for the Token-messages that are used for the intercommunication between the Petri Nets federates through RTI. With regards to (2) the federates are time-regulating and time-constrained and dealing with both kinds of messages HLA provides: the TSO and RO messages. The look-ahead computation has been made according to the position of the scheduled transition in the Petri Nets federate. The look-ahead was the critical point in dealing with simulation of stochastic Petri Nets models as we have shown in the results discussion.
We apparently have not gained any Speed-up compared to the sequential simulation. This may be due to the small look-ahead. Moreover inherent insufficient accuracy of numerical computations of look-ahead in case of non-deterministic models may cause causality violations in the order of execution of events. We experienced that mechanisms provided by HLA do not sufficiently support stochastic behaviors. We consider it as an important field of future development of HLA mechanisms. We are working on the application of optimistic time management, which in fact does not introduce any numerical accuracy problems. Speed-up is not the primary aim of HLA, however the advantage of HLA emerges in interoperability between PN simulations and reusability of components.

Acknowledgements

We express our gratitude to the ZK-FNK (Zentrale Kommission für Forschungsplanung und wissenschaftliche Nachwuchs) of the university of Bremen for providing fund for the project FNK-Nr. 03/182/5 whose results are used for this paper and to TZI-BISS (Technologie-Zentrum Informatik Bremer Institute for Secure Systems) for sponsoring the project leader to attend the Advanced Course on Petri (ACPN 96) Nets in Dagstuhl 1996. We would like to thank Konan Kouakou Dieu-Donné, a graduate student, for his contribution to this paper.

References

[1] Alois Ferscha: Parallel and distributed Simulation of discrete event Systems in Parallel and distributed computing Handbook; edited by Albert Y. H. Zomaya McGraw-Hill 1996 pp. 1003-1041

[2] Homepage for Petri Nets: http://www.aut.utt.ro/~mappy/petri/home.html

[3] Homepage for Petri Nets: http://www.daimi.aau.dk/~petrinet/

[4] Falko Bause, Pieter S. Kritzinger: Stochastic Petri Nets: An introduction to the theorie Vieweg Verlag 1996

[5] Frederick Kuhl, Richard Weatherly, Judith Dahmann: Creating Computer Simulation Systems: An introduction to the High Level Architecture, Prentice Hall PTR, 1999

[6] Giovanni Chiola and Alois Ferscha: Distributed Simualtion of Timed Petri Nets: Exploiting the Net Structure to Obtain Efficiency, in: M. Ajmone Marsan – editor, Application and Theory of Petri Nets, Volume 691 of Lecture Notes in Computer Science pp. 146-165, Berlin, Springer Verlag 1993

[7] Horst Mehl: Methoden verteilter Simulation; Vieweg Verlag Wiesbaden, 1994

[8] James L. Peterson: Petri Net Theory and the Modeling of Systems; Prentice-Hall, Inc. Englewood Cliffs, New Jersey 07632, 1981

[9] Jerry Banks, John S. Carson II, Barry L. Nelson: Discrete-Event System Simulation second edition Prentice-Hall, Inc. Upper Saddle River, New Jersey, 1996

[10] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelii, G. Franceschinis: Modeling with

generalized stochastic Petri Nets John Wiley & Sons, 1996

[11] Richard M. Fujimoto: Parallel discrete event simulation in communication of the ACM October 1990 Vol. 33 No. 10 pp. 30-50

[12] Richard M. Fujimoto: Time Management in the High Level Architecture in Simulation, Vol. 71, No. 6 pp 388-400 December 1998

[13] Richard M. Fujimoto: Parallel and Distributed Simulation Systems, John Wiley & Sons, inc. 1999

[14] Tadao Murata: Petri Nets: Properties, Analysis and Applications in Proceedings of the IEEE, Vol. 77, No. 4 April 1989

[15] Homepage for HLA http://hla.dmso.mil/hla

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37271

AIAA-2000-4397

FIDELITY CONSIDERATIONS FOR CIVIL AVIATION DISTRIBUTED SIMULATIONS

Manfred Roza, M.Sc.
Delft University of Technology, Faculty of Aerospace Engineering
Kluyverweg 1, 2629 HS, Delft, The Netherlands
e-mail: z.c.roza@lr.tudelft.nl

Jeroen Voogd, Ph.D.
TNO Physics and Electronics Laboratory
P.O. Box 96864, 2509 JG, The Hague,
The Netherlands
e-mail: voogd@fel.tno.nl

Paul van Gool, Ph.D.
MimoTech
4065 Foothill Road, Apt. F, Santa Barbara,
CA 93110
e-mail: pvangool@mimotech.com

ABSTRACT: *The technical capability to network together simulators for al kinds of applications has already been proven. There are, however, still many unresolved issues regarding the ability of simulations to inter-operate in a logically meaningful manner. One of the major concerns in development and validation of distributed simulations is the capability to qualify and quantify the 'over-all' simulation fidelity. In order to properly apply distributed simulations to civil aviation applications, such capabilities are mandatory. Therefore, the Delft University of Technology, in co-operation with the Netherlands Organization for Applied Scientific Research, started a fundamental research program on fidelity assessment methodologies for distributed simulations. This paper presents the results of a literature survey into fidelity research, and our approach to managing simulation fidelity. Amongst other things, the survey showed a lack of formal definitions and objective fidelity quantification practices. Here a preliminary fidelity theory is developed, which underlies our evolving 'Fidelity Management Overlay' (FiMO) process model. The FiMO provides a structured approach for the specification of fidelity characterizations, their quantification where possible, and the transformation of fidelity characterizations during the whole development trajectory of HLA-based simulations. The theory is illustrated with examples from a recently developed distributed 'Free Flight' simulation.*

INTRODUCTION

Fidelity is an intrinsic element of any simulation, one which all simulation developers and users have to deal with one way or the other. The concept of fidelity is an apparent vehicle in properly assessing the validity and credibility of simulation results. Furthermore, fidelity is found to be one of the main cost-drives of any model or simulation development. As a general rule, the higher the fidelity the more time and resource consuming the simulation development is. Thus, being able to state what level of fidelity is exactly required avoids unnecessary investments superfluous simulation components and unusable simulation. Despite these observations and the enormous advancements in simulation hardware and software, the ability to characterize and quantify the level of simulation fidelity is still a largely uncultivated area.

Until a few years ago their was no real need for such methods because simulation was used on a smaller scale and primarily developed in-house to tackle some minor aspects of a specific problem. However, the simulation community starts to realize that the current fidelity practices can no longer fulfil the demanding requirements of today's simulation applications.[3][14] The increasing dependency on simulation results, the greater complexity and more frequent reuse of simulations across different application domains causes this desire for new and more comprehensive fidelity methods. With the advent of modern distributed simulation technology, such as the US DoD Modeling & Simulation (DMSO) High Level Architecture (HLA), a whole range of fidelity concerns have been expressed that need to be addressed before the full potential of distributed simulations can be utilized. [3][12][14] This has caused a renewed interest in the development of more comprehensive methods for quantification and usage of fidelity in the context of HLA.

Distributed simulation research has primarily focussed on the technical interoperability of simulations, solving problems regarding the capability of simulations to physically interconnect via a certain communication infrastructure and effectively exchange data in accordance with a set of rules, data-formats and interface specification. HLA has resolved most of the problems concerning the technical capability to network together simulations and its concept is now proven by many DoD applications. Demonstration of technical interoperability, however, is a necessary but not a

Copyright © 2000 by Delft University of Technology. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

sufficient condition to guarantee a credible and valid distributed simulation. If there is no way to determine the credibility or validity of a simulation, decisions based on results of that simulation become questionable. This is of vital importance when applying distributed simulations to safety-critical applications, such as civil aviation applications.
What is lacking, are methods for assessing substantive interoperability.[14] Substantive interoperability focuses on the ability of simulations to inter-operate in a logically meaningful manner up to a level of fidelity that satisfies the simulation objectives and needs. A major issue in this regard, is thus the capability of characterizing and quantifying the fidelity of distributed simulations. Currently, there seems to be no widely accepted methodologies available for the fidelity characterization and measurement of distributed simulations. In order to apply HLA and distributed simulations properly to civil aviation applications such methods are mandatory. Therefore, the Delft University of Technology (DUT), in co-operation with the Netherlands Organization for Applied Scientific Research (TNO), started a fundamental research program to fidelity assessment of distributed simulations.

This paper discusses the results of this on-going research and starts with a summary and analysis of previous published ideas and issues regarding the subject of fidelity characterization and quantification. The paper then describes the basics of a preliminary fidelity theory we developed based on the synthesis of ideas from previous related research efforts and ideas resulting from our own experience. This theory makes it possible to define and measure fidelity in a more formal manner. However, it does not yet provide generally applicable practical guidelines to help simulation developers manage all fidelity decisions they have to make during the development and validation of complex distributed simulations. Therefore, the remainder of this paper continues with a presentation of the development and application our fidelity management process model, called FiMO. This process model is a structured approach for the specification of fidelity characterizations, their quantification where possible, and the transformation of fidelity characterizations during the whole development trajectory of HLA-based distributed simulations.

SURVEY OF EXISTING FIDELITY THEORY AND PRACTICE

In the past decades much work has been published on the subject of simulation fidelity. This section summarizes the most important results our extensive literature study to fidelity research. [13]

GENERAL OBSERVATIONS

The literature on fidelity indicates that the simulation community agrees that fidelity measurement is based on the comparison between reality, or some abstraction of reality, and the simulated representation of this reality.

However, there exist many, different and often contradicting, viewpoints on what the fidelity concept exactly comprises. This causes serious confusion among simulation developers and users, and resulted in the lack of widely accepted methodologies available for characterization and measurement of fidelity. The two basic reasons for this are:

1. Almost all researchers define and use their own terminology for fidelity related concepts, which hampers communication between different researchers.
2. Current simulation practice is an art form and at best an inexact science lacking fundamental and commonly accepted theory in which fidelity can be understood and characterized. Only one, recently published, exhaustive modeling and simulation theory is available.[15]

THE FITNESS FOR PURPOSE CONFUSION

Historically, the question of 'how much realism or fidelity is good enough for our needs' has been the main driver behind fidelity research. This question is the prime reason why people have been wrongly using the term fidelity as a synonym for fitness for purpose.[11] Because simulations are abstract representations of some part of reality, the answer to this question states an important set of requirements for simulation systems, which should guide its development. This resulted in many fidelity definitions and concepts, which incorporate application specific aspects. Clearly, such fidelity constructs are not workable for the whole simulation community and tend to assign different levels of realism for the same simulation when used in another application. Obviously, the simulation doesn't change in such case, nor its level of realism. It is therefore better to separate a formal fidelity definition from application specific aspects. In this way fidelity becomes an absolute measure of simulation realism instead of the often-used relative measure, which have been causing the pointless fidelity discussions.

FIDELITY QUANTIFICATION

Various types of metrics for quantifying simulation fidelity are found in literature. Basically, all these fidelity metrics are rather subjective and qualitative. Analysis of the available literature shows that there is a community wide recognition that subjective adjectives (low, medium, high), which express fidelity in qualitative terms can no longer fulfil the current simulation requirements. What is needed are

quantitative fidelity metrics, concepts and methods to measure the fidelity of simulations in an objective manner, in order to develop simulations that produces the desired and reliable results against acceptable costs in both time and money.

Existing fidelity metrics can be classified in singular dimensionless and complex multi-dimensional metrics. Each of these has its pros and cons.[12][13] A singular over-all fidelity metric may have some utility when a simulation must be selected from equivalent simulations. It provides a tool for a quick first selection. But for making final selection additional fidelity knowledge is required Example of such metrics are the JAA flight simulator classifications (A, B, C, D).[6] A major drawback of summarizing metrics is that they hide the sources of the deviation from reality. Literature illustrates that describing the simulation realism is a complex issue involving many simulation characteristics that are hard, if not impossible, to capture in a meaningful single dimensionless metric. The advantage of comprehensive fidelity descriptions is that they make fidelity an useful development tool. A major draw back of these complex fidelity descriptions is the large amount of data that must be processed, making development more time consuming and expensive. The development of tools to automate such fidelity assessment methods may fully or partially overcome these objections.

FIDELITY ASSEMENT FOR TRAINING SIMULATORS

Traditionally, fidelity research and experience has been a practice well cultivated in the training simulation domain, especially in the pilot training area.[1][6] Most of this research has been, and still is, focussed on the human perception of the entity behavior and representation and how to stimulate the human sensory systems in order to give the human operator the impression that he or she is in the real entity.[4] This type of research always considers fidelity solely in the context of unitary simulators. For unitary simulators, which involve humans, these human-machine interface aspects of fidelity are probably the most important ones for judging the simulator suitability for a certain training purpose but are certainly the only ones.[13]

When several human-in-the-loop simulators are interconnected in a distributed simulation environment (e.g. team training), each simulator must interface with the other simulators. Then the simulator must provide a certain level of fidelity in the human-machine interface and in the interface with the distributed environment. These two 'fidelity' boundaries are connected to each other through the internal structure of the simulator (entity system models, terrain database, mock up, visual system, etc.). This introduces extra complexity and problems in characterizing the over-all fidelity of a complete distributed simulation compared to non-human-in-the-loop complete distributed simulations, which is an issue not really touched upon by existing fidelity literature. Solutions to these fidelity aspects are of vital importance to achieve substantial inter-operation of human-in-the-loop simulators for various purposes such as multi-vehicle air-traffic control and management research.

Some believe that the 'FAA/JAA' simulator requirements for pilot training' fully addresses the whole spectrum of simulation fidelity issues. Indeed, our literature study reveals that these requirements comprise the most mature, practical and proven approach to fidelity developed yet.[13] However, these requirements are specifically tailored for unitary pilot training simulators and based on the assumption that a rich set of real-world data is available. Therefore they are not directly applicable to other simulation applications and simulation architectures, such as distributed simulations.

DISTRUBUTED SIMULATION FIDELITY: THE DIS LEGACY

The literature study yielded only two significant publications regarding the fidelity of distributed simulation, both originating from and referring to DIS based simulations.

The first publication provides an enumeration of simulation properties and physical hardware components, which need to be considered in assessing the fidelity of simulations.[7] Despite, the incompleteness, DIS orientation and high-level descriptive character of this enumeration, it clearly demonstrates that there is a difference between fidelity assessment of stand-alone or unitary simulators and interacting simulators through a network. More importantly, it can be concluded from this work that distributed simulation fidelity assessment comprises all those properties and activities of unitary simulation fidelity assessment and added to this are fidelity issues related to substantive interoperability

The second publication is the draft IEEE-P1278.5 Standard for Distributed Interactive Simulation: Fidelity Description Requirements, which has been developed on a government and industry initiative.[5] This draft standard primarily comprises an exhaustive enumeration of fidelity characteristics without any guidelines on how to use this taxonomy in practice. So far, literature doesn't provide any indication that this draft standard is widely utilized in the simulation community. Three main reasons for this status are the following. The enumeration approach enforces a single and rigid worldview of models to be used in simulations. The enumeration is based on cataloguing currently existing real-world knowledge and therefore

remains always incomplete or insufficient for unforeseen usage. Finally, IEEE-P1278 only addresses DIS-based real-time training simulation applications with the focus on the military problem domain. It's obvious that the IEEE fidelity standard cannot fulfil the needs of today's HLA-based simulations, because HLA aims at simulations for a much broader application and problem domain, often used to evaluate non-existing systems.

TODAY'S FIDELITY RESEARCH

Literature shows that the Simulation Interoperability Standards Organization (SISO) is currently supporting fidelity research via an international fidelity study group (Fidelity-ISG).[3] This SISO fidelity study group follows a scientific approach in seeking a fidelity theory that is generic, formal and robust enough to be applied across multiple M&S application and problem domains. To fill the void of a proper and common agreed simulation terminology they have developed an initial glossary of fidelity related terms.[3] The formal definition for fidelity provided by this glossary is the following: *'The degree to which a model or simulation reproduces the state and behavior of a real world object or perception of a real world object, feature, condition or standard in a measurable or perceivable manner'*. Furthermore, the Fidelity-ISG has developed a conceptual fidelity framework, which outlines several basic concepts to define fidelity and there semantic relationship. Our practical experience shows that the current state of this framework can be characterized as experimental and that it has a high abstraction level, which makes it hard to understand and to apply in practice.[10]

SIMULATION FIDELITY THEORY: A SYNTHESIS

This chapter presents the fidelity theory and framework developed by TNO/TUD, based on the synthesis of results from the literature survey, the recent Fidelity-ISG results and our own research efforts into fidelity.

FIDELITY THEOREMS

The fidelity framework is based on the following developed fidelity theorems:[12][13]

1. Fidelity of a model or simulation is qualified and quantified by an enumeration of various multidimensional and multifaceted metrics for all model or simulation aspects that characterize its degree of realism.
2. Fidelity is an intrinsic and absolute property of a model or simulation and therefore its metrics must not be mixed with problem and application dependent factors.
3. Fidelity is qualified and quantified with respect to a fidelity referent, which is a codified body of knowledge of the real-world counterpart the model or simulation attempts to represent.
4. Fidelity quantification has always a level of uncertainty, which decreases with increasing knowledge about the real-world counterpart a model or simulation tries to represent.
5. Comparison of the fidelity levels of models or simulations for the same real-world counterpart only makes sense when the fidelity specification of each model or simulation is based on the same predefined set of fidelity metrics and measured against the same fidelity referent.

All these theorems are consistent with the fidelity definition in the FDM-ISG glossary. It is primarily the failure to understand or comply with theorems 2, 3 and 4 in the past, which has lead to endless discussions of fidelity as a quest to some kind of a wholly-grail that doesn't exist. A fidelity theory can never give the answer to 'how much realism is good enough' because this varies with the purpose of the simulation, it can only provide the language and tools to support such relative judgements.

FIDELITY DESCRIPTIVE CONCEPTS

Fidelity is characterized and defined by the next seven descriptive concepts (*Figure 1*), which are in accordance with theorems 1 and 2. *Detail* is considered as the major building block for abstracting aspects of the real world. It is a designated part (entity, system, components, object, process, etc.) with certain representational and behavioral characteristics that can be perceived through a set of observable properties. The degree of detail used in a simulation or model is what is called resolution. *Resolution* describes what aspects of reality a model or simulation represents and what aspects have been left out. Resolution is thus the roughest form of measuring the difference between reality and the simulated representation of this reality.

The difference between how the value of the observable properties of each real-world aspect detail vary overtime and measure the difference with its reflecting part in the simulation is often referred as property (parameter or 'state' variable) *error* or *accuracy*. Accuracy is actually the inverse of error.[3] What aspects of reality and with what level of accuracy they are or can be represented by a simulation depends on how these aspects are internally realized in terms of used models (mathematical representations, algorithms and logic), data-sets, software and hardware components (network capabilities, visual system characteristics etc.).

The fidelity concepts of sensitivity and precision are qualifiers for the quality of the internal realization and determine the magnitude of the errors of the observable properties. The concept of *precision* is a

measure how meticulously computational processes are performed by the internal structure of a model or simulation. Precision thus expresses errors due to limitations of solely the internal representation. *Sensitivity* describes the effects of uncertainties and imperfections of external stimulus (input variables) and internal parameters (data values) on the error magnitude of the detail's observable properties.

In real-life, time is a fundamental mechanism underlying our perception and description of how the real-world evolves, i.e. capturing changes over a certain period of duration and ordering mechanism for the occurrences of certain changes (causal order). Simulations itself are by definition dynamic processes, which try to represent systems whose state and characteristics change overtime. Characterizing time aspects and how a model or simulation handles time is described by the concept of *timing*. Time arrangements in a model or simulation are a major cause for errors in the observable properties.

The last fidelity concept is *capacity*. Capacity is a concept associated with the performance characteristics or capabilities of the simulation hardware and software implementation (CPU, network, memory, operating system, simulation environment architecture etc.), and the number of instances of detail that can be represented simultaneously given a certain time performance constraint (real-time constraint, time frame size).



Figure 1: Fidelity Descriptive Concepts Graphic

FIDELITY REFERENT CONCEPT

Intuitively, the most conceptually right form of fidelity measurement is by gathering information from reality and comparing it with data obtained from the simulated counterpart. Simulations are, however, often used to study behavior of existing systems under conditions, which cannot be performed in real-life because of safety constraints etc. Furthermore, simulations are heavily used to investigate systems that partially or fully do not exist. For such purposes there is no or very few observed data available. This may lead to a false assumption that fidelity cannot be measured in these cases. Due to the fact that it is practically impossible to know every thing about reality, some error and uncertainty will always be part of our perceptions and descriptions of reality, even in the cases when there is a rich data set available. Another notion in this regard is that comparisons of simulation data with observed real-world data will only state how well a simulation is capable of replicating these data, not how well a simulation represents behavior of the same system, for which no such data is available. These notions are captured by fidelity theorem 4.

A pragmatic solution to these problems and to make fidelity a workable concept is the introduction of a referent structure. As stated in theorem 3, fidelity of a model or simulation is measured against a referent instead of reality itself. A *fidelity referent* is a formal specification of all knowledge about reality plus indicators to determine the uncertainty levels and quality of this knowledge to judge the confidence level of this referent data. The movement of uncertainties to solely the referent introduces a level of indirection, which enables fidelity specification for any model or simulation, regardless whether it represents a material reality (existing systems) or imagined reality (non-existing systems). In this way fidelity can be used as a beneficial tool for describing and comparing the capabilities of models and simulation. Furthermore, the inextricable question associated with fidelity of 'how much realism is good enough' can then be more transparently defined in terms of *fidelity requirements*, which specifies the tolerated deviation of the simulated reality from the data in the selected referent. The judgement of the validity and credibility of the simulation results then yields the assessment whether the used referent has an acceptable level of uncertainty, suitable for the intended purpose of the simulation and that the simulation execution remains within the defined tolerances with respect to referent data. Fidelity referents that are suitable for a specific application are designated as *normative*. Referents that are community wide accepted for a specific application or problem domain are designated as *authoritative* real-world descriptions. This approach allows effective de-coupling of fidelity metrics from problem and application dependencies as stated by theorem 2. In this regard the JAA/FAA standards can be considered as a combination of a set of fidelity requirements for specific training purposes, and an authoritative referent for pilot training simulators.

SOME PRACTICAL FIDELITY METRICS

For each of the seven descriptive fidelity concepts it is possible to give a set of more formal properties or metrics that allow fidelity measurement of a model or simulation against a fidelity referent. Since the goal of this paper is not to provide or deduce a complete

list of metrics, only the most elementary metrics will be discussed as used in a recently developed distributed 'Free Flight' simulation (FASE).[8] Four types of *detail* sets can be specified:

- *Real-world aspect or object detail.* Example: an airspace system comprises ATC control stations, airports, surveillance radar, Airbus 320 aircraft, Boeing 757 aircraft etc.
- *Real-world aspect behavioral detail.* Example: a Boeing 777 has engine dynamics, flight dynamics, structural dynamics etc.
- *Real-world aspect observable property detail.* Example: the aeroelastic wing dynamics of a Boeing 777 are observed by the wingtip deflection amplitude, wingtip rotation angle, number of bending modes etc.
- *Real-world Aspect interrelationship detail.* Example: a Boeing 747 consists of 4 engines and each engine respectively consists of a low-pressure and high-pressure compressors etc. Another example of interrelationship type is, a fuel system interacts with an engine by means of a fuel flow.

With this approach to abstract reality in terms of detail it is possible to define *resolution* metrics. A first metric used is *detail scope difference*, which describes the difference between the set of details mentioned in the fidelity referent and the set of details contained in the model or simulation. The part of a model or simulation describing the set of details (D_{ms}) should always be a subset of the fidelity referent detail set (D_{ref}). Therefore in a mathematical formulation this metric can be expressed by set theory as being the subtraction of D_{ms} from D_{ref}. This is an intuitive metric since for judging suitability and validity of a model or simulation it is important to known what aspects of reality haven't been modeled. Application of this metric to the FASE application reveals that the FASE airspace model doesn't take into account light-weight aircraft such as Cessna 172 etc compared to all aircraft operating in the current European airspace.[8] Whether a simulator represents certain motion-ques experienced by a pilot in the true aircraft can be specified in terms of detail scope.

A second resolution metric is the *relative detail scope size*, which is the number of details in D_{ms} divided by the number of details in D_{ref}. This metric is of less information contents than the *difference in scope metric* but provide quick insight in the comprehensiveness of a model of simulation. The FASE simulation model for instance contains only 60% of the current existing aircraft types.

Detail multiplicity difference is a third resolution metric, which specifies for each single detail in D_{ms}, the number of duplicates or detail instances compared to the ones in D_{ref}. This is a useful metric because for all identified real-world aspects there often exist many instances of the same aspect in a complex system. For instance, in any real airspace system there fly more than one instance of B747-400 aircraft. Knowing such difference is needed for judging the credibility of certain results obtained by the FASE simulation.

The last resolution metric discussed here is *aspect decomposition depth difference*, which describes the levels of object detail decomposition that has been implemented by the model or simulation compared to the fidelity referent. *Aspect decomposition depth* thus indicates at what level the interactive behavior between separate aspects is aggregated to a single comprising object with an overall behavior. Aggregation of aspects usually involves assumptions and approximations regarding the interactive behavior of the separate parts. This may result in obfuscating or even a loss of certain interactive behavior and their observable properties in the overall represented object detail i.e. loss of some realism that might be of someone's value. In the FASE simulation for example, the computer generated air-traffic aircraft model utilizes an aggregated flight dynamic and engine dynamics model with only a summarized thrust vector property in the center of gravity to represent the aircraft engine behavior. While the used B747-200 pilot-in-loop simulator represents the engine dynamics by separate models and, therefore can represent behavior like differential thrust and high-pressure turbine speed.

Error is a general term for quantifying the difference between a certain value and a correct value.[3] Resolution is thus also a form of error. Therefore, error metrics, discussed next, refer to the errors in the modeled or simulated observable property values compared to the data values in the referent. There exist many ways to measure property errors, depending on the nature of the property.[12] For parametric properties, such as surveillance radar positions etc., the following commonly used error metrics can be used: the measured difference or the absolute value of the measured difference between the value in the referent and the modeled or simulated ones. For time-dependent properties these include, various norm metrics, stochastic metrics like mean and standard deviation, frequency contents metrics, combinations thereof, etc. For example, the difference of a simulated and an actual aircraft pitch angle during a phugoid involves assessing its error in at least three ways: phase shift, damping and eigen-frequency. Proper evaluation of time-dependent properties requires the consideration of the conditions, sample rate and time period for which the fidelity referent has been constructed and simulated property data has been generated.

In reality all aspect behavior and events occur in sidereal time[3] any deviation in time advancement in a simulation is thus a deviation from reality. The *time advancement representation* metric is the first metric

for the fidelity concept of timing, which specifies if the simulation time[3] advances as *real-time* or *non real-time*. Sidereal time advancement is continuous while computer simulated time advancement is made in discrete steps. How far the simulated time deviates from the continuous character of the sidereal time is specified by the *timeframe size* between two successive simulated points of time (T_{i+1} - T_i) of each real-world aspect.

Time advancement rate is a third metric for timing fidelity, which specifies how many simulated time advancements are made during a sidereal time period of one second. *Cumulative simulated time* equals the multiplication of timeframe size and advancement rate, and is a metric for how well non real-time representations simulated time keeps pace with the sidereal time.

The last timing metric discussed here is *causal event ordering difference*, which measures the difference between ordered sequences of state changes and events between simulated real-world aspects and their state changes over some time interval and the causal order as specified by the fidelity referent. An example of this could occur when for instance a model of surveillance radar system sends a secondary radar pulse to an aircraft, which in return replies with a transponder pulse. If for some implementation reasons this transponder sends it's reply pulse after a too long time interval, it might be possible that the radar antenna azimuth angle has changed so much, the radar model ignores the reply. The result of this is that the aircraft incorrectly disappears from the radar screen, because of a wrong cause.

A FIDELITY REFERENT FORMAT

The exact fidelity referent content is not part of a fidelity theory (theorem 2). However, to enable proper fidelity measurement of a model or simulation, fidelity referents need to have a well defined structure, which specifies the real-world knowledge in terms of the discussed fidelity descriptive concepts and provide handles to assess its confidence level. Furthermore, referent's content evolves when new data or insights about reality become available, so referents should be maintainable. From a high-level the developed fidelity referent format comprises the following basic elements:

1. *Referent Identification Section:* This section provides unique referent ID, a version control mechanism and the point of contact information of the organization, which is managing the referent.
2. *Referent Applicability Section:* This section gives a summary description of the referent contents, the referent application and problem domain and authority status within these domains. Any known usage of this referent is listed here as well.
3. *Referent Developer and Validation Agent Section:* This section references the individuals who developed or validated the referent and describes their specific role and tasks in these efforts.
4. *Referent Knowledge Sources Section:* This section gives a full reference to all knowledge sources that have been utilized in the development of the referent.
5. *Real-World Structural Properties Section:* This section defines the structure and interrelationships of all details in the referent both in graphical form as well as a textual format. Furthermore, it describes interactive behavior over time between the separate details, utilizing an UML like use-case and interaction diagram format.
6. *Real-World Parametric and Behavioral Data Section:* This section provides the numerical data values, where possible, for each detail's observable properties. It provides input/output behavioral data or time base series of how a property evolves over-time and time independent or parametric property data. Both parts have a full reference to the origin of the data, including the conditions for which the data values hold, and how the data has been constructed (actual measured, computer generated data etc.)

Section 1 contains the information needed for maintaining the referent. Sections 2 through 4 basically provide the indicators to assess the quality or confidence level and applicability of the referent. The last two sections do contain the actual real-world knowledge descriptions, which are utilized to measure the model or simulation fidelity.

FIDELITY IN HLA-BASED SIMULATION DEVELOPMENT

Being able to define and to provide metrics for quantifying simulation fidelity in a more objective manner is only one part of the fidelity issue. Determining the relationship of fidelity to the entire development process of HLA-based distributed simulation or federation, including validation is an equally important aspect. Only when it is understood how fidelity should be applied and managed, the full potential of fidelity characterization and measurement in developing more credible and economical federations can be exploited.[12] How the current HLA technology addresses fidelity is presented in next sections.

HLA: INTEROPERABILITY AND FIDELITY

One of the primary HLA goals is promoting and achieving interoperability of geographically distributed simulation facilities. As stated in the introduction, interoperability issues can be divided in two areas of concern, technical and substantive.[14]

However, this must not lead to the misconception of having two unrelated problems that can be solved independently. Substantive and technical interoperability problems are closely related. It is the substantive interoperability, the ability of simulation facilities to operate together in a logically meaningful manner up to a level of fidelity that satisfies the simulation needs, we are looking for when developing distributed simulations using HLA. The HLA technology itself provides the technical solution to enable interaction of simulations in standardized fashion that is needed to achieve the desired substantive interoperability. [8] This technical interoperability is implemented by the Run-time Infrastructure (RTI), which is the physical communication layer for interconnecting the separate simulations. Since this RTI is part of the simulation infrastructure it's performance, such as latencies, can and will affect the fidelity of the abstract world created by the interconnected simulations. Latencies are a major cause of causality problems.

The greatest strength of HLA, having a RTI and associated services to operate independent of the simulation contents, allows simulation community-wide application of the architecture. However, it is also a major cause for fidelity problems. HLA prescribes that a Federation Object Model (FOM) and a Simulation Object Model (SOM) must be created. A FOM describes all commonly agreed data to be shared and intercommunicated between all federates during the federation execution. A SOM describes the information a simulation can produce and requires when participating in a federation. In order to execute the RTI it is necessary to feed it with a so-called FED-file, which is a sub-set of the FOM data, to set-up its internal data structure. Having defined a FOM doesn't give any guarantee that the actual data values of the FOM data are correct since the RTI doesn't provide a mechanism for checking this. Having defined a SOM for all simulations is also no guarantee for proper fidelity, since it doesn't specify how the application code (models) of each simulation processes or produces the SOM data. Finite state representations of each simulation introduce for instance errors in the data produced by a federate, which propagate through the whole federation and effect other federates independent of the specified accuracy levels in the FOM/SOMs. In short: having a FOM and SOM is no guarantee for proper substantive interoperability of all simulations in a federation.

Another aspect for HLA federations is achieving a common understanding of reality between all simulations. Especially, when more than one simulation, represents exactly the same real-world aspect in its internal model, it is necessary to establish a suitable correlation between the multiple instances. Failure to do so severely effects the fidelity of the distributed simulation. These problems mostly apply to the environmental representations such as terrain and atmosphere. For instance in air-traffic scenario it might well be possible that one of the operating aircraft simulators is capable of viewing another aircraft, because clouds are not modeled, while others may not see the other aircraft because of the presence of clouds in their visual.

HLA: FEDEP MODEL AND FIDELITY

DMSO defines a process for building HLA-based simulations: the Federation Development and Execution Process (FEDEP).[2] It defines a set of sequenced activities, which support the development of HLA-based simulations. Six steps are defined by the current FEDEP description: Federation Objectives Definition, Conceptual Model Development, Federation Design, Federation Development, Federation Integration and Test, and Federation Execution and Analysis (*Figure 3*). A more detailed discussion on the FEDEP contents and practical application can be found in an accompanying paper.[8]

Fidelity effects all stages of simulation development. Practical experience shows that the FEDEP certainly does address some aspects related to fidelity but in an implicit manner, which is incomplete and not specific enough.[10][11] The prime cause of this is that the FEDEP process is heavily oriented towards achieving technical interoperability of simulations in terms of required HLA products. It fails to fully answer and support the substantial interoperability issue. Therefore, following the FEDEP is no guarantee for achieving sensible and realistic HLA-based simulations.

Being able to characterize the reality that needs to be represented in a simulation, i.e. fidelity, is a key element to achieve this substantive interoperability. Therefore, what is needed in this regard is a systematic and structured way to help guide the simulation developers in efficiently characterizing and quantifying fidelity aspects, and managing all fidelity decisions and considerations in the FEDEP. A transition to a more fidelity driven approach is expected to result in a better and more efficient way to obtain substantive interoperability of simulations.

FIDELITY MANAGEMENT OVERLAY (FiMO)

The above mentioned considerations regarding fidelity and HLA-based simulations are the driving force behind the ongoing development of the Fidelity Management Overlay (FiMO) for the FEDEP model discussed in this section

FiMO: BASIC FRAMEWORK

The FiMO provides a structured approach for addressing the numerous fidelity characterization and measurement issues that occur in the activities of

each simulation development stage, and how fidelity descriptions are transformed when transitioning from one stage to the next. The basic framework comprises 5 major, iterative, fidelity assessment activities, which maps to the FEDEP model (*Figure 2*).[12][13]

The FiMO starts in the first two stages of the FEDEP, with the specification of the required levels of overall fidelity of the whole networked HLA-compliant simulations or federation. These fidelity requirements are deduced from the federation needs and objectives using the FiMO Fidelity Requirements Specification (FiReS) method. These federation fidelity requirements are the blueprint for estimating and allocating the needed fidelity of the separate simulations (federates) during the design stage. These fidelity requirements concern the real-world dimensions (which aircraft types to be represented, tolerance levels for aircraft properties, etc.) and the implementation dimensions of fidelity. Fidelity implementation dimensions describe the physical, functional and performance characteristics of a specific hard and software elements, which determine and impact the final federation fidelity (visual system, processor type, latencies, etc.).

Based on the federate fidelity requirements and the federation development constraints, suitable existing simulations can be selected or modified to be reused, or new simulations can be developed. The federation development may reveal some unexpected difficulties in the actual implementation of the design requiring modifications to the design or limitations of the implementation. Addressing and documenting the effects of these modifications and limitations on the final fidelity for both the federate and federation are part of the FiMO process. In the integration and testing stage of the FEDEP, the FiMO successively checks if the fidelity requirements of the federates and federation are achieved for the current implementation. During the federation execution by the user fidelity characteristics are monitored and analyzed to assure that the federation indeed fulfils the fidelity requirements. Finally, all information regarding the fidelity of the developed federation and federates must be archived to enable future reuse.

FIDELITY REQUIREMENTS SPECIFICATION (FiReS) METHOD

The importance of developing appropriate fidelity-based simulation requirements cannot be underestimated in this framework. Fidelity requirements are the mechanism to link the application independent fidelity concepts to the federation specific needs or objectives in terms of specifying the appropriate level of fidelity required for these simulation objectives. Elicitation of needs and objectives is the hardest part of simulation requirements engineering.[11] However, failing to elicit all true needs and objectives will result in a set of fidelity requirements that will not be appropriate for the federation purpose. The FiReS method is an iterative process for capturing true needs and objectives, and maps these to fidelity requirements.[11] FiReS comprises the following three main activities:

1. Elicitation, analysis and negotiation of user needs or objectives, possible fidelity requirements related information and development constraints. The FiReS utilizes fourteen sets of basic questioner lists plus associated data sheets to develop a formal commonly agreed problem and need statement.
2. Creation of the federation fidelity requirements. This comprises the selection or development of a suitable fidelity referent with the help of Subject Matter Experts (SMEs). Determination of how far the simulated world may deviate from the fidelity referent in terms of required resolution, detail property tolerances and timing properties. Selection of metrics and measurement methods to measure whether the actual simulation meets these requirements. Furthermore, the simulated operational context or scenarios requirements must be specified here.

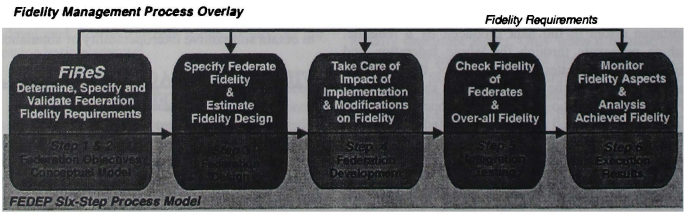


Figure 2: FEDEP and Fidelity Management Overlay

3. Specifying the fidelity requirements for the conceptual model implemented by the federation. This comprises specifying the requirements for the mathematical algorithms (parameter values, underlying assumption and structural form), hardware requirements including hardware-in-loop considerations, human behavior representation and human interface fidelity in terms of cue requirements etc. This specification should be complemented by 'ideal' SOMs and timing approach for each anticipated federate.[10][11]

VALIDATION & VERIFICATION USING THE FiMO

The FiMO structure implicitly contains a process for fidelity-based Verification and Validation (V&V). Using the Fidelity study group definitions for fidelity and validation, validation can expressed in terms of fidelity, as *'the process of determining whether the level of fidelity of a model or simulation is sufficient from the perspective the intended purpose of the model or simulation'*. This last definition is the one that is used in the FiMO for capturing the validation aspects of simulation development. It fully complies with the fidelity theorem 2 and the fidelity referent concept.

Fidelity-based V&V activities in the first two stages of the FEDEP comprises the assessment whether the specified fidelity referent and requirements created with the FiReS are sufficient for achieving the federation needs and objectives. When the fidelity referent and requirements specification are accepted by the federation user and SMEs, they are considered to be valid. The federation design & development activities can only commence effectively when this credibility has been established first. Fidelity V&V activities in these two steps focus on verifying whether the fidelity requirements are properly translated to an appropriate design and implementation of the federation. In step five of the FEDEP, fidelity-based V&V focuses on measuring the fidelity characteristics as stated in the fidelity requirements for both the federates separately and the federation as a whole, and check if they comply with these requirements. The final activity of a fidelity-based V&V process is establishing, in co-operation with the federation user and SMEs, that the in stage five measured and checked level of federation fidelity indeed fits the federation purpose.

CONCLUSIONS

This paper shows that due to the increasing use of complex distributed simulations in today's society there is an increasing interest for characterizing and quantifying fidelity. Especially, in safety critical applications such as civil aviation applications fidelity is an essential metric for being able to properly assess the validity and credibility of the simulation results. The literature study demonstrates confusion about what fidelity exactly is. The survey shows a lack of formal definitions and objective fidelity quantification practices for distributed simulations as well as unitary simulations.

Based on these findings a foundation for a fidelity theory is presented. This fidelity theory builds upon five theorems, seven descriptive fidelity concepts and a fidelity referent concept that formally define the principles of fidelity assessment. The major fidelity metrics for several descriptive fidelity concepts are given along with practical examples. An analysis of the current state of the HLA technology shows some fidelity pitfalls and demonstrates that it doesn't properly support fidelity characterization and quantification in its development process. To deal with this weakness a fidelity management process is proposed, the Fidelity Management Overlay (FiMO). The FiMO provides a structured approach for the specification of fidelity characterizations, their quantification where possible, and the transformation of fidelity characterizations during the whole development trajectory of HLA-based distributed simulations, including validation.

Future research shall focus on the expansion and refinement of the presented fidelity theory and FiMO by means of practical application in the development of distributed civil aviation simulations. These refinements will consider fidelity in the context of a broader modeling and simulation theory. Furthermore, the development of a set of tools are foreseen to support the simulation developer in understanding, analyzing and managing the large amount of complex fidelity-related data handled in the FiMO.

ACKNOWLEDGEMENTS

The authors would like to thank the Simulation Interoperability Standards Organization (SISO) Fidelity study group for their encouragement and support in preparing this manuscript and especially its chair Mr. D.C. Gross of The Boeing Company and S.Y. Harmon of Zetetix. The views of this paper are those of the authors and should not be construed to represent views of the SISO Fidelity study group or any of its members. Web-site:
http://siso.sc.ist.ucf.edu/stdsdev/fidelity/index.htm

REFERENCES

[1] AGARD. *'Fidelity of Simulation for Pilot Training'*. Advisory Report No. 159, Advisory Group for Aerospace Research & Development. Neuilly sur Seine, 1980.

[2] Defense Modeling and Simulation Office (DMSO), *'Federation Development and Execution Process (FEDEP) Model, version 1.5'*. Department of Defense: December 1999.

[3] Gross D.C., et. al., 'Report from the Fidelity Definition and Metrics ISG'. *1999 Spring Simulation Interoperability Workshop*, March 1999, Paper 167.

[4] Hays R.T., M.J. Singer. *'Simulation Fidelity in Training System Design: Bridging the Gap between Reality and Training'*. Springer-Verlag New York Inc, 1989.

[5] IEEE, *'Standard for Distributed Interactive Simulation - Fidelity Description Requirements'*. IEEE-1278.5, 1995.

[6] Joint Aviation Authorities, *'Joint Aviation Requirements: JAR-STD 1A Aeroplane Flight Simulators'*. JAA-HQ: April 1997.

[7] Lane N.E., E.A. Alluisi. *'Fidelity and Validity in Distributed Interactive Simulations: Questions and Answers'*. Institute for Defense Analyses. Alexandria (VA), November 1992

[8] Roza M. and P. van Gool. 'Simulation Free Flight in HLA Federations'. *2000 AIAA Modeling & Simulation Conference*, Denver: August 2000, Paper AIAA-2000-4399.

[9] Roza Z.C., D.C. Gross and S.Y. Harmon. 'Report Out of the Fidelity Experimentation ISG'. *2000 Spring Simulation Interoperability Workshop*, March 2000, Paper 151.

[10] Roza M., 'Experiences in (Ab)Using the FDM-ISG Fidelity Framework'. *99 Fall Simulation Interoperability Workshop*, September 99, Non-Paper Presentation for the FEX-ISG.

[11] Roza M., J. Voogd, H. Jense and P. van Gool, 'Fidelity Requirements Specification: A Process Oriented View'. *99 Fall Simulation Interoperability Workshop*, September 99, Paper 032.

[12] Roza M., P. van Gool and H. Jense, 'A Fidelity Management Process Overlay onto the FEDEP Model'. *98 Fall Simulation Interoperability Workshop*, September 98, Paper 083.

[13] Roza Z.C., *'Simulation Fidelity: A preliminary study to fidelity assessment methodologies for HLA based distributed simulations'*. TNO-Report FEL-98-I311. TNO Physics and Electronics Laboratory: The Hague, December 1998.

[14] Turrell C., et. al., 'HLA and Beyond: Interoperability Challenges'. *99 Fall Simulation Interoperability Workshop*, September 99, Paper 073.

[15] Zeigler B., et al., *'Theory of Modeling and Simulation, 2nd Edition'*. Academic Press, Inc.: January 2000

AUTHOR BIOGRAPHIES

MANFRED ROZA is a member of the scientific staff in the Aerospace Engineering, Control and Simulation Division of the Delft University of Technology (DUT). He holds a M.Sc. (1997) in Aerospace Engineering from the same university. He has experience in HLA-based distributed simulation development and implementation. Since 1998, he is one of the lead-team members of the SISO Fidelity Study Groups involved in the development and application of a fidelity taxonomy and fidelity framework for HLA-based distributed simulations. Currently, he is pursuing a Ph.D. in simulation science from the same university in close cooperation with the Simulation Division of TNO-FEL.

JEROEN VOOGD is a member of the scientific staff in the Command & Control and Simulation Division at TNO-FEL. He holds a Ph.D. (1998) in Computational Physics from the University of Amsterdam. He has experience in modeling and simulating biophysical systems on parallel and distributed computing platforms. Currently he is involved in projects on virtual reality, joint forces operations and connecting C4I infrastructure to simulators.

PAUL VAN GOOL is a Senior Software Engineer with MimoTech Software. He received his M.Sc. (1992) and Ph.D. (1997) in Aeronautical Engineering from the Delft University of Technology in the Netherlands. He has experience in the mathematical modeling of fixed- and rotary-wing aircraft for real-time simulation and the design and implementation of real-time distributed simulation systems. His current activities include developing distributed systems for large scale messaging services. His HLA activities range from developing several federations to developing object-oriented frameworks to facilitate federation development. His research interests include parallel and distributed computing. He focuses primarily on programming paradigms and environments for tightly coupled computers and distributed simulation / modeling systems.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37272

AIAA-2000-4399

SIMULATING FREE FLIGHT IN HLA FEDERATIONS

Manfred Roza, M.Sc.
Stephen Fer.
Delft University of Technology, Faculty of Aerospace Engineering
Kluyverweg 1, 2629 HS, Delft, The Netherlands
e-mail: z.c.roza@lr.tudelft.nl

Paul van Gool, Ph.D.
MimoTech
4065 Foothill Road, Apt. F, Santa Barbara, CA 93110
e-mail: pvangool@mimotech.com

ABSTRACT: *Modernization of existing airspace systems requires various new architecture plans to be developed and evaluated for their improvements in air-transportation efficiency and safety. The high level of automation and increased data communication in the interaction between the various distributed airspace components used in these proposed architectures cause enormous complexity in the overall system behavior, which is hard to evaluate with single unitary simulations during engineering design. Therefore, the Delft University of Technology set the objective to develop a distributed simulation environment for evaluating future ATC/ATM concepts, as part of their future airspace research programs. In order to support the architectural definition of this new Future Airspace Simulation Environment (FASE), a prototype distributed simulation has been developed to evaluate various aspects of aircraft operation and operational procedures that are associated with the implementation of the 'Free Flight' paradigm. This paper specifically focuses on the practical experience with the 'High Level Architecture' technology, development process and tools that are utilized in this prototype work. Based on the first experimental results and lessons-learned from this prototype work, the validity and possible benefits of using HLA for research, development and engineering (RD&E) applications in the ATC/ATM domain is evaluated.*

INTRODUCTION

There is a general consensus between the aviation authorities and the aviation industry on the enormous potential for time and resource savings associated with future flights that are subject to less Air Traffic Control (ATC) restrictions. Removal of restrictions represents a move toward a 'Free Flight' environment, in which pilots have more freedom in determining their flight-path in real-time. In support of the 'Free Flight' paradigm, authorities and industry need to invest billions of dollars to develop, and introduce new Communication, Navigation, Surveillance (CNS/FANS), airborne systems and Air Traffic Management (ATM) technologies into their airspace systems. In order to modernize the various airspace systems, an architecture plan needs to be developed and evaluated for its improvements in air-transportation efficiency and safety. It is expected that with the deployment of these new CNS/ATM capabilities, airspace users will benefit from improved services, such as more optimized cruise trajectories and altitudes, and more efficient surface traffic operations. The change in controlling air traffic across airspace sectors, gives pilots more unprecedented decision making authority in selecting the routes, speeds and altitudes to their destination airport, and adjusting them throughout the flight execution.

Copyright © 2000 by Delft University of Technology. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

The 'Free Flight' paradigm and new CNS/ ATM capabilities impose a high level of automation and increased data communication between a large set of distributed airspace system components. This causes the interaction between the various airspace components and the overall system behavior to be highly complex. And therefore hard to evaluate with separate unitary flight and ATC/ATM simulations during engineering design. The Delft University of Technology Department of Aerospace Engineering (DUT-AE) has the objective to develop a distributed simulation environment, in which various types of flight and ATC/ATM simulations can inter-operate in order to create a set of coherent representative future airspace systems of varying complexity. This Future Airspace Simulation Environment (FASE) enables the evaluation of the feasibility, efficiency and safety of new airspace components and concepts within an integrated operational context.

To support the definition of the FASE architecture, a prototype distributed 'Free Flight' simulation is currently under development. This paper presents the first results and lessons-learned of this FASE

prototype research. More specifically, this paper focuses on the application of the distributed simulation technology used in this effort: the High Level Architecture (HLA) with its supporting, development process and tools. Based on the FASE prototype experience, the usability, benefits and problems of using HLA for research, development and engineering (RD&E) applications in the ATC/ATM domain is evaluated. Finally a discussion is presented on of the future development of the FASE and related research on distributed simulation technology.

DELFT AEROSPACE ATC & ATM RESEARCH

The airspace architecture currently in use consists of a fixed set of airways and airspace sectors with centralized ground-based air-traffic control (ATC) facilities. This architecture provides not enough flexibility and capacity to be able to efficiently handle the expected increase of air-traffic. To address this problem the DUT-AE initiated a series of research programs under the name En-Route Airspace 2020, Airport 2020, and Flight Deck 2020. The main objective of these programs is the development of new airspace architectures and associated technology, and the evaluation of their improvements in air-transportation efficiency (lesser delays, cost-effective flight profile, minimal operational constraints) and capacity, without compromising agreed levels of safety and environmental pollution (noise, exhaust gases). Within these research programs there is a close cooperation with other institutes such as Eurocontrol, the institute for Motion, Simulation Navigation (SIMONA) technologies and TNO Physics and Electronics Laboratory (TNO-FEL).

En-Route Airspace 2020

The En-Route Airspace 2020 (ERA-2020) program focuses primarily on the en-route flight phase in higher airspace regions and implementation of the 'Free Flight' paradigm. The major concepts of Free Flight are[7][12]:

- The transition from (navigation) point to point airways to a direct route airspace system in which aircraft operators have more autonomy in determining the most optimal route.
- Transfer of separation assurance and conflict resolution from ground controllers to the aircraft systems and pilots.

Fundamental research questions addressed in the ERA-2020 program include the evaluation and analysis of combining of full-autonomic aircraft and direct-routing concepts to create an en-route airspace architecture, which represents a self-regulating multi-agent system. Furthermore, the ERA-2020 project includes studying new airborne separation assurance systems (ASAS), flight rules and CNS/FANS. Another important aspect which is addressed is what role human operators, both pilot and ground-controllers, should be given in these new airspace systems, and what levels of automation, situational awareness and decision-making authority are needed to fulfil this role properly.

Airport 2020

One of the effects of direct-routing in an en-route airspace is the transfer of traffic-flow bottlenecks to the airspace area directly surrounding airports, called the terminal areas (TMA). Free Flight concepts are not directly applicable here, because of limited runway capacity, high air-traffic density, safety regulations and inconvenience for surrounding populated areas. Therefore, different ATC/ATM concepts need to be developed in which the responsibility for safe and efficient air-to-air separation is a distributed task between aircraft pilots and ground-controllers. This is often called the 'Freer Flight'.[4] The main objective of the Airport 2020 program is to develop and evaluate new ATC/ATM concepts to realize a smooth and efficient (maximum capacity less delays) transition between the Free Flight en-route airspace and a Freer Flight TMA. New concepts and technology under consideration in this effort include curved and low-power approaches, power-constrained departures, air-traffic transition systems logic, and 4D navigation.

Flight Deck 2020

The trends towards free and freer flight airspace architectures demand the improvement and (re)design of existing avionics and flight deck human machine interfaces. The Flight Deck 2020 program aims at improving avionics systems and flight deck interfaces in such a way that they truly support the flight crew in the control and monitoring tasks of current and future air navigation and airspace systems developed in the other 2020 projects. The current Flight Deck 2020 research involves the study of 4D flight path oriented primary flight displays, cognitive traffic displays (situational awareness), energy monitoring systems, and intelligent and multi-modal flight deck interfaces.

FUTURE AIRSPACE SIMULATION ENVIROMENT (FASE)

This section discusses the rationale, development approach and envisioned architecture of the Future Airspace Simulation Environment (FASE) currently under development within DUT-AE.

RATIONALE AND OBJECTIVES

Most of the research in all three programs is conducted by means of a mixture of different types of simulations and simulators. However, these programs require similar modeling and simulation efforts of

existing and future airspace components (CNS, FANS, aircraft, ATM systems, ATC controller stations, etc.), and simulations in a natural environment. To enhance the development of the individual required models and simulations it is necessary to promote reuse and integration of already existing Modeling and Simulation (M&S) capabilities. As argued in the introduction, to be able to evaluate the improvements of new ATC/ATM concepts and airspace component technology (CNS, etc.) on the overall airspace system behavior it is necessary to integrate all individually developed airspace component simulations to create a representative operational context. Furthermore, it is also desirable to have the capability of integrating ATC/ATM simulation facilities of the participating external partners, which may be located at different sites. These requirements are the drivers for the development of the new DUT-AE simulation infrastructure, 'Future Airspace Simulation Environment (FASE)'.

ARCHITECTURE AND DEVELOPMENT APPROACH

A distributed simulation architecture approach is chosen because of the scalability, flexibility and re-configurability of such a distributed simulation environment. This allows various types of simulations to inter-operate in order to create a set of coherent representative airspace systems of varying complexity. Such a distributed architecture promotes easy exchange, reuse and interoperation of simulations, simulation components and models developed in all three programs, which decreases simulation development time and increases the ease of integration of research efforts. The types of inter-operating simulations within a distributed architecture can range from simple air-traffic flow generating simulations, desktop flight simulators, pseudo-pilot stations, to research air-traffic control simulators and full flight simulators. Furthermore, the physical capability of such a distributed simulation environment to connect geographical distributed ATC/ATM simulation facilities within a single co-ordinated exercise offers a powerful tool for future collaborative ATC/ATM research with the external partners.

There exist much software and hardware technology to implement a distributed simulation environment (DIS, CORBA, etc). For several reasons the High Level Architecture (HLA) of the US Defense Modeling & Simulation Office is chosen for the FASE infrastructure. First, our recent experience with HLA in the SIMULTAAN project demonstrated that much of the HLA capabilities suite the requirements of the FASE.[1] The resulting knowledge and software from this project enables the incorporation of the rapid prototyping paradigm to the selected incremental development approach for this FASE. Furthermore, the first experiences with HLA for ATC/ATM applications are encouraging.[8] Other considerations include, the HLA is mandatory for all US governmental simulations, HLA is currently targeting for IEEE standardization and there is a large supporting community for the HLA. As a result much information on research, software and development tools is available at low or no expense. It is expected that many future aerospace simulations will have HLA capabilities. Such future aerospace simulations may therefore easily be integrated within the FASE.

AVAILABLE AND DESIRED COMPONENTS

Both the DUT-AE and other partners have already many simulation facilities available that will be adapted and integrated in the FASE architecture. For representing the airborne component of the future airspace the DUT-AE has it's own fixed based research flight simulator available, TNO offers a similar type of research flight simulator and the SIMONA institute will incorporate their new 6DOF full flight research simulator into the FASE. Eurocontrol provides STANS a TCP/IP based multi-platform research simulation for current ATC/ATM systems. It comprises a pseudo-pilot station used for air-traffic generation, simulation of current CNS and ATC/ATM systems, and traffic controller working positions.[5] Furthermore, Eurocontrol offers several future CNS simulations and an aircraft performance data-base (BADA).[6]

To be able to fully address the TUD-AE research programs also several new FASE components are desirable such as, virtual workbenches for traffic controller stations, computer generated air-traffic simulators, communication and navigation satellite system simulations and ATM facility simulations. Furthermore, hardware-in-the-loop, such as live radar data or research aircraft, is also expected to be needed for this research.

HIGH LEVEL ARCHITECTURE: AN OVERVIEW

Most research on the interoperability of simulations has been and still is being performed in the military application domain. This research has resulted in the development of the High Level Architecture (HLA) by the US Defense Modeling & Simulation Office (DMSO). The High Level Architecture (HLA) provides a common simulation architecture, which promotes the interoperability and reuse of physically distributed simulation facilities and components. The major objective of the HLA is the execution of simulation scenarios using distributed and independently designed facilities in a unified simulation environment. Since these characteristics are also highly desirable for civil aviation simulations, there is a large interest for using HLA in

the context of distributed (global) research on the future airspace system.

MAJOR CONCEPT AND TERMINOLOGY

A typical HLA application is referred to as a *federation*. It consists of a number of physically distributed simulations called *federates*, which communicate with each other through a physical interconnection layer known as the *runtime infrastructure* (RTI). Federates can consist of entire simulation facilities, simulation components or support facilities. Federates can be independently designed but to properly participate in a federation they must obey certain design rules and be able to share all federation relevant data in a standard HLA format using a standardized interface functionality or protocol. Federates which can fulfil these requirements are designated HLA-compliant.

HLA BUILDING BLOCKS

The formal definition of the HLA comprises three main elements: the HLA Rules, the HLA Object Model Template (HLA-OMT) and the HLA Interface Specification:

- HLA Rules: a set of ten basic rules that together describe the general principles of the HLA, relating how federates and federations are constructed. Five rules apply to the design of federates and five rules organize the design process of the Federation.
- HLA Interface Specification: a definition of the functional interactions between federates and the HLA Runtime Infrastructure (RTI). The interface specification defines six communication services associated with federation management, object management, time management, declaration management, ownership management and data distribution management.
- HLA Object Model Template (OMT): a common template for specifying and documenting federation object models (FOM) and simulation object models (SOM). A FOM describes all data to be shared and intercommunicated between all federates during the execution of a federation. All data communication as defined in the FOM goes through the RTI. A SOM describes the information a federate can make available and requires when participating in a federation.

RUN TIME ARCHITECTURE

The RTI is a distributed system and interconnection layer that facilitates the communication over the simulation network between all federates in the federation (*Figure 1*). The RTI implementation comprises basically three main components:

- Federation Executive (FedExec): a global process of each separate federation execution, which controls joining and resigning of federates, and assures proper FOM data exchange.
- RTI Executive (RtiExec): a global process for the creation, destruction, location, and connection to federation executions i.e. FedExec.
- RTI federate library (libRTI): a software library implementing the RTI interfaces services. When this library code is invoked it creates a process called local RTI component (LRC), which handles all data and network communication between the federate application code and the federation.

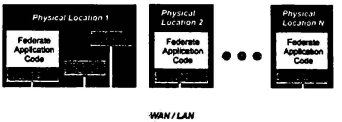


Figure 1: RTI Architecture

To set up and run a federation execution the RTI requires two configuration files. The first is the federation execution data (FED) file containing the listings of the FOM data. The second file is the RTI initialization data (RID) file containing information to control and tune the RTI software, such as network parameters.

FEDERATION DEVELOPMENT AND EXECUTION PROCESS

The physically distributed character of a HLA-based simulation environment, its interoperability and reuse objectives and the possibility to design and implement components independently typically requires the use of advanced engineering management operations. Therefore, the HLA is accompanied by the Federation Development and Execution Process (FEDEP) model.[2] This FEDEP is a scalable system engineering process that provides a high-level framework to organize the whole life-cycle of a federation, from development activities to actual executions of the federation. The FEDEP model comprises 6 basic iterative steps (*Figure 2*). Recently, the FEDEP model is extended with a set of checklists: the FEDEP Checklists.[3] These checklists provide additional, more detailed, activities to assist the developer in properly achieving the required products and results, as specified by the FEDEP model.

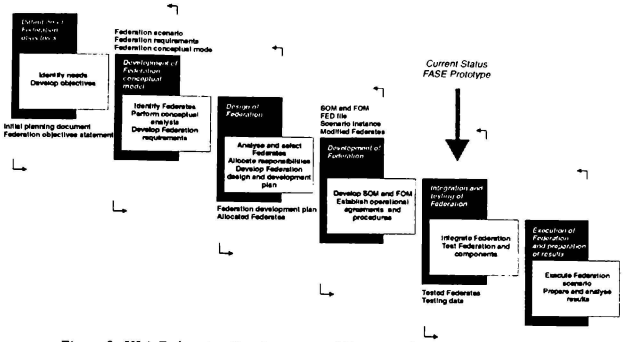


Figure 2: HLA Federation Development and Execution Process (FEDEP) Model

FASE PROTOTYPE OBJECTIVES

The first version of the Future Airspace Simulation Environment (FASE) is now reaching its completion. Currently, we are in the integration and testing stage of the FEDEP. This first version is developed to support the architectural and requirement definition of the final simulation environment. It should, however, already provide sufficient functionality to conduct some basic research on the 'Free Flight' paradigm in the context of the ERA-2020 program. Summarized, the FASE prototype objectives are:

- Gaining additional experience with the HLA and more particularly the use of the FEDEP and available federation development tools.
- Evaluation of HLA as a tool for distributed ATC/ATM simulation applications, and identifying possible limitations and problems.
- The resulting federation should represent a basic future airspace implementation suitable for studying various aspects of aircraft operation and operational procedures that are associated with the implementation of the 'Free Flight' paradigm.
- Refining and elicitation of requirements for the FASE architectural definition.

FASE PROTOTYPE DEVELOPMENT: A FEDEP MODEL USE-CASE

This section discusses how the FEDEP model, checklist and other available HLA tools have been used in the development of the FASE prototype federation. It should be noted that the FEDEP model is executed as an iterative process but for convenience the results are presented in step order. FEDEP step six is not discussed in much detail because at the time of writing the paper the FASE prototype development is in step five of the FEDEP.

STEP 1: DEFINITION OF THE FEDERATION OBJECTIVES

The first step of the FEDEP is the determination of the objectives that should be accomplished by the federation. This step identifies the main problem to be tackled by the federation and the identification of the requirements for the HLA application. Furthermore it includes the precise determination of the federation top level characteristics, the users/research preferences and expectations, the operational context and constraints of the federation development. Besides providing this indicative list of possible resulting products, the FEDEP and checklists do not provide a thorough mechanism to deduce these products. To fill this void the Fidelity Requirements Specification (FiReS) method has been used. This is a structured process for developing proper federation requirements.[11]

Problem and need statement

The input to the FEDEP and FiReS are the program objectives and available resources. For the FASE prototype these have been stated in a formal document, which has been summarized in the previous section. In this document the available resources are an inventory of available personnel, hardware, software, development tools etc. The following formal problem & need statement, in summarized form, has been developed for this federation by using the FiReS:

- *Problem Context:* The developed federation is only used by the DUT-EA for their own ERA-2020 program to study the Free Flight paradigm.
- *Problem Domain:* ATC/ATM of future airspace systems.
- *Application Domain:* Research, Development and Engineering (RD&E) applications.

- *Concrete Problems to be Addressed by the Federation* (in order of descending priority):

  1. Airborne Separation Assurance Systems (ASAS) conflict detection and resolution algorithms evaluation and optimization for both single and multi-aircraft conflicts.
  2. Evaluation of Cockpit Displays of Traffic Information (CDTI) for supporting the pilot's task within the separation assurance process.
  3. Studying the limitations of a Free Flight en-route airspace in terms of airspace capacity i.e. traffic density vs. safety.
  4. Studying the traffic-controllers role in a Free Flight airspace and how to maintain proper situational awareness.

For each of these concrete problems the needed simulated data and usage have been used to determine what aspects of simulated reality are of interest and to define the minimal set of data that should be collected during federation execution.

Federation objectives

The federation objectives for the FASE prototype are stated in the form of a set of executable scenarios for each of the concrete problems. The executable scenarios define what types of scenarios have to be executed by the federation to obtain the desired results, and they identify what model or simulation parameters should be tunable by the user. Furthermore, they identify the major systems of interest and operational context that need to be simulated. For example for problem one and two the major executable scenarios comprises a set of conflict geometries between two aircraft approaching each other under various operational flight phases such as one aircraft in cruis-climb intersecting the flight path of the second aircraft which is in level cruise flight. For each of the identified conflict geometries the federation execution initial and termination conditions have been defined: the initial state of each aircraft, flight-plan and atmospheric conditions. The federation execution is terminated when the conflicts have been resolved and both aircraft are back on their initial flight path. Tunable parameters include ASAS algorithm parameters, displayed traffic information, ADS-B transceiver range and broadcast rate.

The major FASE prototype development constraints and required resources, in summarized form, are:

1. Completion data by 31 July.
2. Availability and usage of already existing flight dynamic models, CNS models and ASAS logic.
3. HLA tools to use and evaluate DMSO Object Model Development Tool, Federation Execution Planners Workbook Tool, Federation Management Tool and Data Collection Tool.
4. SIMULTAAN RCI software for Windows NT.
5. Usage of the DMSO RTI-1.3v6 for Windows NT because of the SIMULTAAN RCI constraint.
6. All software will be implemented in object-oriented C++ code.
7. Only 4 PIII 600 desktop PCs are available for this first stage interconnected in a LAN. Most legacy simulation facilities of Eurocontrol, SIMONA, and TNO are not yet available.

STEP 2: DEVELOPMENT OF THE FEDERATION CONCEPTUAL MODEL

The second step of the FEDEP is the development of a federation conceptual model. As in the previous stage the FEDEP provides a set of indicative tasks and products, that includes the identification of participating simulation components, the specification of the federate tasks, behaviors and interactions in the context of the federation, a conceptual analysis of the federation and precise set of federation requirements. Again the FEDEP fails to provide a good definition and mechanism to develop and document a conceptual model. Therefore, we adopted a previously published more scientific approach for iterative conceptual model development.[9] According to this publication a conceptual model should address the following aspects: simulation context, simulation elements and simulation concept.

Already in the first stage of the FEDEP the problem and application domain have been identified. They serve as the basis for FASE simulation context development, which is primarily a large body of literature publications and Internet references on both the subject of future airspace concepts as well as existing ATC/ATM research simulation environments. To clarify and complement this database of knowledge sources, additional information has been obtained from FASE participant's domain-experts. All knowledge has been documented according the TUD-AE fidelity referent format.[11]

Next the simulation elements should be identified. Basically, this means that the required real-world aspects and level of aggregation is selected from the referent structure. In this selection process the priority order of the concrete problems was useful in making trade-off decisions between the represented real-world complexity and the federation feasibility within the development constraints. This resulted for instance in very highly aggregated traffic controller communication and surveillance components while the aircraft are decomposed into several sub-system components. The resulting set of simulation elements and their interaction have been documented using UML object model diagrams (*Figure 3*).

Finally the simulation concept has been defined. This comprises the algorithm selection, assumptions made

for implementing the simulation elements and specifying conceptual design decisions. A legacy non-linear 6-DOF B747-200 model has been chosen to accommodate the required pilot-in-loop aircraft simulation for the CDTI evaluation objective of the FASE prototype. To represent larger air-traffic airspace densities without having too many participating pilots and simulation facilities, computer generated aircraft (CGA) are simulated using the Eurocontrol BADA aircraft performance model, which models about 90% of the current air-traffic.[6] The atmosphere is simulated using the international standard atmosphere model complemented by a wind vector profile specified in several altitude bands. Furthermore, the atmospheric effects on the transmission of SSR pulses, transponder pulses, radio messages and ADS-B message have been neglected for this FASE prototype. Radio communication is of less importance for the FASE prototype and therefore implemented by standard textual phraseology. This enables the CGA to implement some human-behavior model that is capable of translating messages to flight management system (FMS) commands and sending replies to both human controllers and pilots within the same federation execution. In case no CGA are part of the federation scenario a separate voice communication line can be used for radio communication. The ASAS detection/resolution module has a generic structure to allow for easy implementation of newly developed ASAS algorithms. In order to facilitate federation testing, a default implementation for this module is provided using already available ASAS algorithms.[12]

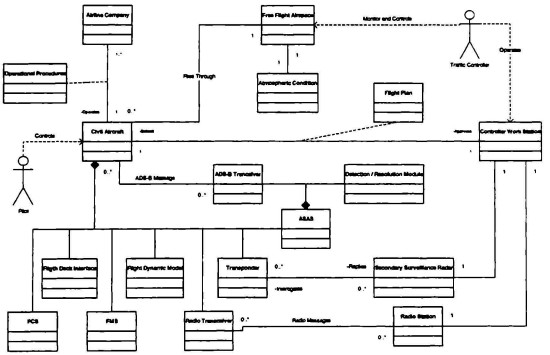


Figure 3: Excerpt of the FASE Prototype Federation Conceptual Model

STEP 3: DESIGN OF THE FEDERATION

The third FEDEP step focuses on the detailed design of the federation and the federates. This includes the identification, evaluation and selection of the simulation participants based on their abilities to meet the federation objectives and requirements. Once this is done, the responsibilities within the federation are to be allocated to the concerned participants. The FEDEP checklist for this step does provide some useful characteristics that were helpful in perform these activities.

First a list of required federate types has been compiled. FASE objectives 1 and 3 require a federate type capable a generating several levels of air-traffic density. Objective 2 requires at least two pilot-in-loop aircraft simulating federate. Objective 4 requires that one controller working position federate should be available. It may be desirable to have a federate for simulating complex CNS system dynamic. However, the FASE prototype conceptual model (*Figure 3*) shows that only one SSR radar and one radio-station simulation is needed and that the models used for these systems are rather straightforward. Since the SSR radar and radio-station are the tools of the controller, it has been chosen to integrate them within the controller federate type. Proper federation execution and data-collection requires runtime-tools that are capable to participate in the FASE federation. The required functionality for each desired federate has been determined from the conceptual model. This resulted in an initial high level SOM for each federate and a FOM for the FASE prototype federation (See next section for more details).

Based on the desired federate types and their required functionality, all candidate simulations of all

participants have been evaluated for their suitability. Due to the availability constraints, time constraints, and the anticipated difficulties in adapting the available complex legacy simulations to achieve the FASE prototype objectives, it has been chosen to develop each federate from scratch. These legacy simulations will be adapted and integrated in the next FASE version. For the execution management and data-collection the DSMO Federation Management Tool (FMT) and Data Collection Tool (DCT) have been selected. Both tools are designed as separate federates capable of joining any kind of federation. *Figure 4* shows the final FASE Prototype architecture including all allocated federate functionality. Since there are only PCs available, the pilot-in-loop flight simulator federate has been designed as a desktop flight simulator using off the shelf game devices for flight control input. The flight-deck interface such as the needed primary flight display, CDTI and out-of-window view, are simulated on a single monitor using OpenGL. For the air-traffic controller federate common keyboard and mouse input devices have been selected. The controller's plan-view display (PVD), electronic data display (EDD) and radio stack are all simulated on a 22" monitor to achieve orderly and well observable displays.

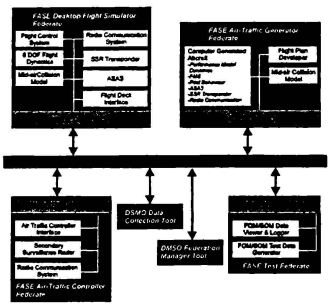


Figure 4: FASE Prototype Federation Architecture

The connection between the federate's application code and the Local RTI Component (LRC) is established by using the SIMULATAAN RCI for Windows NT.[1] This RCI is a middle-ware layer that provides an abstraction layer and API, which shields the application developer from the underlying interoperability standard complexities such as the DIS, HLA or any other dedicated protocol. The SIMULTAAN RCI is based on the HLA RTI 1.3v.X versions.

The current architecture and available 4 PCs provide reasonable federation configuration flexibility, up to six possible configurations. For each federation execution PC four must be used as a host for the executing the RtiExec, FedExec, DMSO FMT and DCT runtime-tools. All three remaining PC's are used for executing the other FASE federates in the following configurations:

1. PC 1, 2 and 3 each executing a single FASE Air-Traffic Generator federate instance to simulate a free-flight airspace with large air-traffic density to support FASE objectives 1 and 3.
2. PC 1, 2 and 3 each executing a single FASE Desktop Flight Simulator federate to study pilot behavior in solving multi-aircraft conflicts in a free-flight airspace (FASE objective 1 and 2). A separate voice communication system is optional using to study effect of voice communication on the conflict solving process.
3. PC 1 FASE Air-traffic Controller federate, PC 2 a single FASE Desktop Flight Simulator federate and PC 3 either a FASE Desktop Flight Simulator or Air-Traffic Generator federate to study the future role of a human controller within a free-flight airspace (FASE objective 4)

STEP 4: DEVELOPMENT OF THE FEDERATION

The fourth step in the FEDEP is the development of the federation. In this step the following three major activities are performed: development of the FOM, achieving additional agreement on initialization procedures, synchronization, data collection procedures and the actual implementation of each designed federate. For this step the FEDEP checklist also defines a large list of useful sub-activities. Experience has shown, however, that most of these activities should already have been touched upon during conceptual modeling and federation design stages since they imply or relate to essential conceptual model and design decisions.

FOM development for the FASE prototype federation involves the refinement and fixation of the initial FOM and federate SOMs developed in previous FEDEP steps. One extension made to the initial FOM is the addition of transmission range and position parameters to all interaction classes in order to facilitate the processing of these interactions by the federate's internal model algorithms. Such parameters are not part of real-world counterparts of these interactions. The final FASE prototype FOM is presented in *Table 1*. During the FOM and SOM development activity the DMSO Object Model Development Tool (OMDT) has been used to validate that all FOM and SOM elements are filled out properly and to create the FED-file. The OMDT provides a spreadsheet like presentation for all FOM/SOM elements, which for larger applications rapidly become disordered. Especially for the object and interaction class structure a graphical UML like representation proved to be desirable. Therefore we developed an extension for the UML standard for

visualizing and expressing FOM, SOM and federate dependencies within standard UML software engineering tools. Another advantage is the transparent and seamless mapping of these HLA elements to the software design and implementation of each federate.

Object Class Structure	
Aircraft (PS)	
Flight_Plan (S)	ICAO_Flight_Plan (PS)
	Experimental_Flight_Plan (PS)
Interaction Class Structure	
ADSB_Message (IR)	
Radio_Message (IR)	
Radar_Puls (N)	Secondary_Radar_Puls (IR)
	Primary_Radar_Puls (N)
Transponder_Puls (N)	Transponder_A_Puls (IR)
	Transponder_C_Puls (IR)
	Transponder_S_Puls (IR)

Table 1: FASE Prototype FOM

The next activity is the actual development of the application cod for each federate and link it to the RCI run-time environment using the automatic code generation facilities of the RCI. The input to the code-generator is the federation FED-file. This generated RCI code shields the application programmer from doing elaborate bookkeeping concerning attribute updates, while making use of the encoding and decoding facilities offered by the RCI run-time environment to communicate attribute and parameter values along the physical network.

STEP 5: INTEGRATION AND TESTING OF THE FEDERATION

The fifth FEDEP step is the integration and evaluation of the federation. This includes the determination of all information required to support the federation execution, the integration of all federation components and the interconnection off all software/hardware and testing each federate separately and federation as a whole.

The first activity performed in this step is filling-out the Federation Execution Planners Workbook (FEPW) using the DMSO FEPW tool. For each executable FASE federation configuration a FEPW has been created. This FEPW comprises a range of tables, documenting which federate executable is running on which computer, computer information and network performance characteristics. Furthermore, it specifies for each federate the size and update/reflect or send/receive rate of FOM specific data during federation execution. For instance the FASE desktop flight simulator sends a new position update with a rate up to 100 [Hz] and sends ADS-B message interactions at a rate similar to broadcast rate of a real ADS-B transmitter (20 to 40 updates per minute). This information is useful for selecting network devices with proper bandwidth and latency. The network connection for the FASE prototype is established with a stand-alone LAN implemented using a dual speed hub. Network latency is therefore negligibly small. Due to constraints in the currently used RCI all federates utilize the non-regulating and non-constraint RTI time mechanism.

Effectively debugging and testing of HLA federations proves to be a difficult task. The approach chosen for the FASE prototype is testing and debugging the federate application code as a stand-alone application first. Next, testing and debugging the HLA interface and interoperability capabilities of each individual federate. For this purpose a FASE test federate has been created capable of stimulating and monitoring the agreed FOM capabilities of each individual federate (*Figure 4*). The FASE desktop flight simulator and air-traffic controller federate already passed this test phase. Both federates are currently being tested in integrated federation configurations 2 and 3 to simulate several free-flight scenarios. The first inspection level verification and validation (V&V) demonstrates the expected federation behavior for these configurations.[9] However more extensive V&V and performance testing is needed to make a final judgement regarding the FASE prototype capability of fulfilling the federation objectives and achieved level of fidelity.

STEP 6: EXECUTION OF THE FEDERATION AND PREPARATION OF RESULTS

The final step in the FEDEP is the execution of the federation, the operational use, the processing of outputs and reporting the results. In this step all participants join in an integrated scenario from which all the relevant federation data and information is collected to evaluate whether the objectives have been met. For the FASE Prototype this yields the execution for doing the desired free-flight research in the context of the ERA-2020 project. This might reveal certain usability deficiencies that will be fixed and used to refine the requirements for the next FASE increment.

LESSONS-LEARNED FROM THE FASE PROTOTYPE WORK

This paper presented the DUT-AE Future Airspace Simulation Environment (FASE) prototype federation development, which is currently reaching its completion. The prototype federation is capable of simulating the major aspects of a Free Flight airspace necessary to support research on various aspects of aircraft operation and operational procedures in such a future airspace.

The first experimental results of the FASE prototype work confirm the conclusions of other research efforts that HLA is a suitable distributed simulation tool for ATC/ATM RD&E applications. However,

some reservation is needed since a more thorough evaluation of the FASE prototype is still in progress. The FASE prototype work demonstrated that HLA is a flexible distributed simulation architecture, which allow interoperation and quick reconfiguration of different types of aircraft and ATC/ATM simulations to create various representative airspace systems. These HLA characteristics offer a powerful tool for future collaborative ATC/ATM research with other research organization. The major difficulty of HLA, experienced in the FASE prototype work is its steep learning curve. Middle-ware technology such as the SIMULTAAN RCI does flatten this learning curve but is most effective in substantively reducing the time needed to create a HLA interface implementation. In this way more time can be spend to the federate application code.

The HLA Federation Development and Execution Process (FEDEP) model and checklists have proved to be a useful *guideline* for the FASE prototype development. Basically, the FEDEP focuses on when and how to achieve the HLA products (FOM, FEPW, etc) needed to create a technical correct and executable federation. We found that the FEDEP doesn't provide sufficient guidelines for developing simulation requirements and conceptual models. Other applied approaches turn out to be more fruitful.[9][11] Another recommendation is to study FEDEP steps in advance since some of its activities should be addressed in an earlier step than they are mentioned in the process. In this way the FEDEP will become a more optimal process tailored for the application at hand. Testing of the FASE prototype proves to be a considerably more difficult and time-consuming task than expected. Careful planning of the federate/federation test phase and using an incremental testing approach is necessary. The available DMSO development tools for HLA utilized in this prototype work (OMDT and FEWP tool) are useful in gaining valid HLA products but are not always very user friendly. Especially, a graphical FOM/SOM representation and integration with software engineering tools is highly desirable. Our in-house developed UML extensions for HLA prove to be an adequate solution for these two problems. The DMSO run-time tools (FMT and DCT) prove to be very useful for general federation execution management and collecting FOM data.

FUTURE RESEARCH

Our future research will first focus on the completion of the FASE prototype federation as outlined in this paper and utilizing the latest release of the RCI for Windows NT developed by TNO-FEL, which supports the RTI-NG1.3v3. A more thorough V&V process and more stringent performance test is scheduled. For the next FASE increments the following extensions and improvements are foreseen:

- Development of a reference FOM for ATC/ATM federations: A FOM and meta-data endorsed by all research partners, which shall be used as a building block for the development and establishment of FOMs and SOMs for new ATC/ATM federations.
- Adaptation of the available legacy simulations of the research partners to the FASE architecture.
- Improvement of the existing FASE prototype federates and creation of new FASE federates such as more complex CNS system and ATM tool simulations.
- Research on and application of HLA RTI ownership management and data distribution management functionality with regard to FASE. Ownership management can be used to transfer an aircraft simulation from an air-traffic generating federate to flight simulator federate and vice versa.

ACKNOWLEDGEMENTS

The authors wish to thank Marco Brasse M.Sc. and Jeroen Voogd Ph.D. of TNO-FEL in the Hague for their guidance and technical support regarding the SIMULTAAN RCI. Furthermore, we acknowledge Eurocontrol Head Quarters for donating their STANS platform and Eurocontrol Experimental Centre for making their BADA model available to our ATC/ATM research programs.

REFERENCES

[1] Brasse M., W. Huiskamp, O. Stroosma. 'A Component Architecture for Federate Development'. *99 Fall Simulation Interoperability Workshop*, September 99, Paper 025.

[2] Defense Modeling and Simulation Office (DMSO), *'Federation Development and Execution Process (FEDEP) Model, version 1.5'*. Department of Defense: December 1999.

[3] Defense Modeling and Simulation Office (DMSO), *'Federation Development and Execution Process (FEDEP) Checklist, version 1.5'*. Department of Defense: December 1999.

[4] Duong V.N., E. Hoffman, J.P. Nicolaon. *'Initial Results of Investigation into Autonomous Aircraft Concept (FEER-1)'*. First USA/Europe ATM R&D Seminar. Saclay, June 17-20, 1997.

[5] Eurocontrol Division DED-2, *'STANS ATC Prototyping Platform, Overview'*. Eurocontrol Agency: Brussels, November 1998.

[6] Eurocontrol Experimental Centre, *'User Manual for the Base of Aircraft Data (BADA), Revision 3.1'*. Eurocontrol France, EEC note No. 25/98. Bretigny-sur-Orge: November 1998.

[7] Hoekstra J.M., R.N.H.W. van Gent, R.J.C. Ruigrok. 'Conceptual design of Free Flight with airborne separation assurance'. *1998 AIAA Modeling & Simulation Conference*, August 1998, Paper AIAA-98-4239.

[8] Meloche P., M.W. Blake, D. Johnson. 'Joint FAA/Army/NASA Federation'. *98 Fall Simulation Interoperability Workshop*, September 1998, Paper 122.

[9] Pace D., 'Simulation Conceptual Model Development'. *2000 Spring Simulation Interoperability Workshop*, March 2000, Paper 033.

[10] Roza M., J. Voogd, H. Jense and P. van Gool, 'Fidelity Requirements Specification: A Process Oriented View'. *99 Fall Simulation Interoperability Workshop*, September 99, Paper 032.

[11] Roza M., J. Voogd and P. van Gool. 'Fidelity Considerations for Distributed Aviation Simulations'. *2000 AIAA Modeling & Simulation Conference*, Denver: August 2000, Paper AIAA-2000-4397.

[12] Schild R., *'Rule Optimization for Airborne Aircraft Separation'*. Ph.D. Dissertation, Technische Universitat Wien, Technische Naturwissenschaftlichen Fakultat. Wien: 26 November 1998.

AUTHOR BIOGRAPHIES

MANFRED ROZA is a member of the scientific staff in the Aerospace Engineering, Control and Simulation Division of the Delft University of Technology (DUT). He holds a M.Sc. (1997) in Aerospace Engineering from the same university. He has experience in the mathematical modeling of aeroelastic aircraft dynamics and aircraft systems for real-time simulation. Furthermore, he has experience in object-oriented software engineering, framework design and HLA based simulation development. He is an established SISO Fidelity Study Group member and participates in the SCS M&S glossary development team. Currently, he is pursuing a Ph.D. in simulation science from the same university in close cooperation with the Simulator Division of TNO-FEL.

STEPHEN FER is a M.Sc. student at the Aerospace Engineering, Control and Simulation division of the Delft University of Technology (DUT). His M.Sc. thesis focuses on the development of a simulation environment for the future civil airspace (FASE) using HLA technology. He has experience in aircraft performance modelling and during his internship at the Eurocontrol headquarters in Brussels, he has gained significant experience in air traffic control and air traffic management (ATC/ATM) simulations.

PAUL VAN GOOL is a Senior Software Engineer with MimoTech Software. He received his M.Sc. (1992) and Ph.D. (1997) in Aeronautical Engineering from the Delft University of Technology in the Netherlands. He has experience in the mathematical modeling of fixed- and rotary-wing aircraft for real-time simulation and the design and implementation of real-time distributed simulation systems. His current activities include developing distributed systems for large scale messaging services. His HLA activities range from developing several federations to developing object-oriented frameworks to facilitate federation development. His research interests include parallel and distributed computing. He focuses primarily on programming paradigms and environments for tightly coupled computers and distributed simulation / modeling systems.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37273

AIAA-2000-4400

Real World Environment Interfaces for Modeling and Simulation

William J. Bezdek *
The Boeing Company
St. Louis Missouri

Thomas A. Halley
The Boeing Company
St. Louis Missouri

Joseph Samocha
The Boeing Company
St. Louis Missouri

Abstract

It is desirable to use object-oriented design techniques and "real world" interfaces to achieve maximum reuse of aerospace models to support system prototyping, development, and testing. Models intended for use in a simulation should be flexible enough to allow them to communicate across distributed networks that minimizes the amount of data sent / received. The architecture should also support models with different levels of detail (fidelity) as well as Distributed Interactive Simulation (DIS) and/or the High Level Architecture (HLA) for distributed simulation. In the past, many simulation models were written in FORTRAN or 'C', were modular, and constructed using "subroutines". Several modular, physics-based models have recently been converted and combined with object oriented simulations for use on PCs and low-cost workstations. A study will be made to see if real world interface requirements can be developed using existing design techniques. A bandwidth vs. fidelity vs. recalculation trade-off will be done to determine the capability to use and switch modeling fidelity during execution. The conclusions will quantify the advantages of using object-oriented design approaches and "real world" interfaces.

Previous Experience

In the past, many simulation models were generally in scientific (e.g., FORTRAN) or software development (e.g., 'C') languages and then extensively modified to comply with the specific requirements of each simulation environment. These models were modular in nature and constructed using subroutines and/ functions to approximate "real world" boundaries: (e.g., the control system, the weapons, the avionics, the engine, etc.).

These were connected using global interfaces without information hiding and often had built-in simulation features, such as:

- the ability to initialize, stop, or hold model state,
- the maximum number of models available were built into array sizes (i.e., the number of weapon stations on an aircraft), and
- key data and phenomenology assumptions were often hard coded into the program.

Other simulations had features built into the models that required a lot of set-up instead of keeping the data separate from the model. In another variation, developers attempted to keep the model code generic, but created input data set structures, which are, in essence, a unique language unto itself.

The government has multiple constructive level models, like Thunder[1], EADSIM[2], Brawler[3], Suppressor[4], DISAMS[5], and ESAMS[6] that have many of the same problems. These models have been described as "Stovepipe"[7] models, in that they don't interact with each other or work together.

Over the years, attempts were made to make these "Stovepipe" models cooperate with each other. The overall approach was developing a unique, specific

* Senior Member

Copyright © 2000 by The Boeing Company. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

spanning architecture that used data parsers, data translators, cradles, and rewriting model code into a called subroutine or function. DIME[8] and MOSAIC[9] are examples of specific architectures for model cooperation. These were the first steps towards flexible, readily linked system, but remain grounded in the non-real time simulation environment.

These simulations are being replaced with new object based modeling and simulation systems, like Joint Warfare System (JWARS)[10], Joint Simulation System (JSIMS)[11], and Joint Modeling and Simulation System (JMASS)[12].

The government and industry have been using DIS[13] for several years to support networking between separately hosted simulations and is in the process of developing and fielding a networking concept called the High Level Architecture (HLA)[14] to expand the capability to link such simulations. HLA will be used over the next few years to enable many such simulations to be linked.

Model Design

The model designer must understand what development and testing activities the components need to be supported and where these activities will take place to maximize reuse of Modeling & Simulation (M&S) models, objects, and object components. This will determine whether one model can support all anticipated activity or whether a family of models, with different fidelity levels and different resource constraints, should be developed. Whenever possible, new or redesigned components should leverage designs available from legacy programs.

In a perfect world, a reuse library would support storing the design or graphical model as well as the source code, and graphical design tools would contain a reverse engineering capability to allow legacy models to be easily updated/converted to the standard format. The graphical design tool would also allow changes to be made to the models at the source code level and automatically have them added back into the graphical model representation. Rational Rose and other tools have that capability; i.e., if another attribute is needed, it can be added to the source code and becomes part of the graphical model. The automatic code generator within the graphical design tool should also create protected areas that allows the model designer to add code segments that become part of the model and are saved when the model is regenerated. JMASS, for example, has that capability called "spin and weave" that allows the designer to add behavior code in special areas that are retained when the Model Component Development Tool (MCDT) regenerates the model.

The Modeling Guidelines support the development and reuse of models to improve affordability. The guidelines recommend that:

- Modeling objects conform to a common Software Structural Model (SSM) to maximize object compatibility and interchangeability without inhibiting model efficiency or model developer creativity,
- Model development be done by domain experts as opposed to software experts only,
- Tools be used to support development of model components which comply with the architectural standards (i.e., developers should not have to worry about the model's target language, only about designing it with visual engineering tools),
- Models be designed for maintainability over an anticipated life cycle of 25 to 30 years,
- The Verification, Validation, and Accreditation (VV&A) process be done in accordance with DoD and Commercial requirements. The test cases and their results show the testing results,
- Environmental modeling be done to support analysis throughout the frequency spectrum (RF, IR, EO, visual),
- Multiple languages be supported (FORTRAN, C, C++, Java, Ada83, and Ada95),
- If existing "legacy" model are used, provide a wrapper to conform to "real world" interfaces and use a procedure call to hide the implementation details,
- Models be developed to support a distributed processing environment,
- A reuse library be established and supported by management to encourage model reuse, and
- Models be stored, as much as possible, with all design details, including requirements, algorithms, documentation, to allow different tools to be used.

Level of Fidelity

Three "levels of fidelity" are recommended to support the anticipated range of simulation applications / environments, including:

(1) Analytic/Constructive models which provide state information and basic signature values for different angular relationships,

(2) Dynamic/Real Time models that have the capability to interact with the on-board avionics and other systems including the pilot. Many manned simulations fall into this category and are

required to cycle in real time while using "real world " interfaces, and

(3) Emulative models that are capable of operating in a detailed measure-countermeasure (e.g., electronic warfare or communications environment. This category tries to provide, as nearly as possible, an exact software replica of the hardware of real system.

The relationship between man and machine is often used relative to model fidelity as shown on Table 1.

Table 1 Man/Machine Interface

Fidelity	Content			
	Man		Machine	
	Sim.	Real	Sim.	Real
Analytic	✓		✓	
Dynamic		✓	✓	
Emulative	✓		✓	✓

For example, the different levels of fidelity for an aircraft are shown on Figure 1.



Figure 1 Model Level of Fidelity

Analytical / Constructive Models

Analytic models are often used in constructive simulations, which require basic information on the model, but in a way that minimizes computer resources, so many models can be instantiated. However, to provide this basic information, the complexity of the models in the constructive simulation can vary widely, dependent on the context. Thunder, as a theater-level simulation, tracks many hundreds of players and, therefore, must aggregate aircraft capabilities into probabilistic representations. Brawler, as a many-on-many air combat simulation, has 6 DOF aircraft aerodynamic performance models, multi-spectral signatures, multiple sensor models and pilot decision logic. DISAMS[15], a one-on-one engagement model contains very detailed model of the threat missile (including missile seeker detection physics and circuit-level guidance and counter-countermeasure logic), the target aircraft signature propagation physics, and models of the environment/terrain itself. In all cases, the analytical simulations can use 10K source lines of code (SLOC) for just one model instantiation and 100K SLOC for the complete simulation.

At the time, building these simulations took months to years to develop. On the other hand, a very detailed 6 DOF aerodynamic performance model may require only as many as 20K SLOC, if it is designed and executed in MATRIXx[16], and is available within two or three days. The user can:

- compute the trim point,
- generate a linear model about the trim point,
- compute the frequency response of the system, and
- perform a gradient search for the coefficients of the equivalent system's transfer function

all within MATRIXx.

Mission software testing can often be adequately accomplished with an equivalent system model represented with as few as 200 SLOC. The 100 to 1 reduction in model size reduces compile and execute times from minutes to seconds. Consequently, a mission software development and testing process that uses the MATRIXx equivalent system model avoids the time and cost associated with independently developing and maintaining an air vehicle model.

Dynamic / Real Time Models

Development / prototyping simulators are normally used to evaluate and test new concepts, including:

- aerodynamics,
- flight controls,
- engines,
- avionics,
- sensors,
- threat tactics, and
- weapons

in real time simulations with project personnel and pilots, and are used to justify modifications for the next iteration or funding cycle of the weapon system. This type of simulation is also used extensively for pilot demonstrations. With the advent of faster processors, the models used for manned simulation are often very close to handling qualities models.

The models being developed need to be able to support and interface with a dynamic executive structure to allow the aircraft, weapons, sensors, visuals, etc. to

communicate with each other. They also need to support "real world" interfaces, so the results of the prototype simulation can easily be the starting point for the software design and coding phase. In addition, it is helpful to use "real world" scenarios and missions that can be developed using a tool, like STAGE[17] during different phases of the development and as a visualization tool to monitor aircraft operation in the environment.

Emulative Models

Emulative models are often designed to operate at very short update intervals to accurately evaluate operations of system elements like radar and electronic countermeasure subsystems.

The emulative model is not normally expected to operate in real time, but must nonetheless retain the same message passing / interface used at the other fidelity levels, with additional data passed between model components to support the higher fidelity calculations within those components. Message passing bandwidth may often increase from approximately 160 variables, 20 times per second to 1250 variables, 200 times per second. This 100 fold increase in message passing would overwhelm an analytic or dynamic model; therefore the emulative model should be run off-line and generate tables which can be used in analytic/dynamic models, which will therefore have traceability back to the detailed emulative model.

Common Interfaces

The way to make models interchangeable is to pay special attention to the interfaces that the models use. In particular, it is important to use inheritance of a common interface for the basic body classes as shown in part on Figure 2.

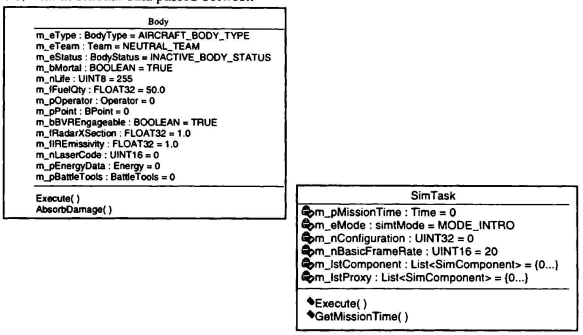


Figure 2 Common Interfaces

The common interfaces can be determined from previous experience with distributed simulations, providing base classes that produce common results, like matrix and vector information including state information for moving bodies, location for targets and fixed objects, etc. The exact representation of the classes needs to be investigated in more detail. As a baseline, the information needed for all models including the constructive model will be listed first. The dynamic/emulative models will append additional messages, if needed. Recognizing that network bandwidth was often a problem, updates were made to DIS to provide additional functionality. The system is called the High Level Architecture (HLA) and is in use for several simulators.

In many cases the data needed for an emulative simulation is compiled and published by the subject matter experts and known as the models are developed. These interfaces may not be extensions of the analytic and dynamic models, but it would be desirable in the future.

The high level architecture (HLA) is a government sponsored system that provides a communications

infrastructure to facilitate interoperability between different simulations and the ability to "federate" existing simulations in order to improve reuse. HLA uses modular simulation components constructed with well-defined interfaces. HLA provides interoperability within simulations, among simulations of a federation, or across functional communities. HLA supports a broad range of functional areas including training, contingency planning, analysis, and acquisition. Applicable simulations involve software only representations, man-in-the-loop simulations, and live components.

Graphical Design Tools

Recently, the use of Ada95 and C++, along with object-oriented design tools and techniques, has had a large effect on reusability and extensibility of simulation models. There are several commercial graphical tools and techniques, like Rational Rose[18], Rose RT, ISI MATRIXx, and i-Logix Rhapsody[19] available to help support object-oriented design by the subject matter experts. In addition, there are several government architectures to help provide reusable, standardized models, like JMASS and others. A typical set of tools used for object-oriented, graphical software development is shown in Figure 3. This typical set includes a/an:

- requirements tool to capture model requirements,
- graphical display tool using open systems,
- graphical object/connection drawing tool,
- automatic code generator,
- configuration management system,
- test case tracking system, and
- debugger.

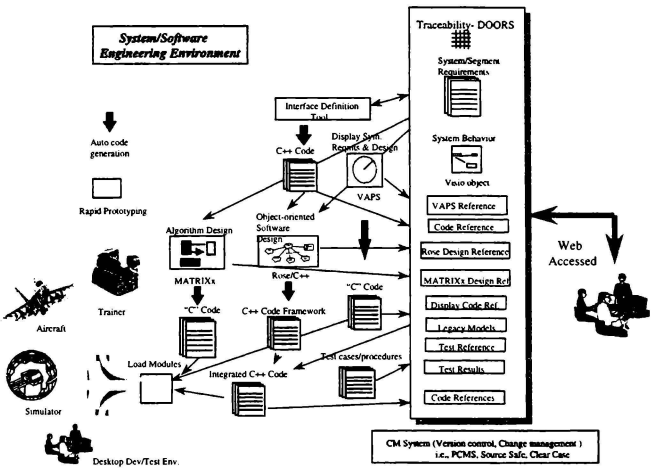


Figure 3 Software Processes and Tools

With graphical tools there is often a cost of ownership, difficulty in model transportability, and the capability to modify a model is significantly impacted if the person wanting to reuse the model does not have the same tool. To provide a more robust model library, it is desirable to store the model design as well as the complete source code for the model, including a description of the model behavior that can readily be used by any tool that supports automatic code generation. Storing the model design also allows models to be independent of programming language and allows different tools to be used with the models.

For example, JMASS provides an intermediate ASCII file called the "description file" with each model that describes the inputs, outputs, attributes, and operations. Rose has an ASCII CAT file with similar, but different, information that goes along with the model file. In addition, the tests used to validate the model should be stored with it to allow it to be retested in the future.

Rational Rose

Rose, one of the most complete commercial, object-oriented, graphical software tools available, was developed by Rational Corporation and is operational on several platforms including PCs and Unix workstations. Rose uses the Unified Modeling Language (UML)[20] developed by Jim Rumbaugh, Grady Booch, and Ivar Jacobson, including a graphical development environment and automatic code generation to allow the design and interfaces, including attributes, to exist in a graphical format along with the state diagram to show behavior.

The developer can:

- build a Use Case diagram which is an example of the procedure for the model,
- develop a Scenario Diagram to define the needed objects and relationships for and between the classes,
- expand the Class Diagram with attributes and operations,
- use the State Diagram to look at the model dynamic operation and the physical file mapping, and
- use the Deployment View to distribute the processes among processors,

The application of the Use Case diagram is being used for requirement analysis and captures the functional requirements of the system. Rose provides a way to then map requirements into an object model. Use Cases can be used to bridge the gap between the analyst, customer, application developer, and help define the tests required.

For the mission used as an example throughout this paper, a commander (actor) selects the target, aircraft type, and pilot for the mission. The pilot selects the weapons desired and flies the mission. On the "enemy" side, the commander, pilot, and SAM controller try to keep the target from being destroyed as shown on Figure 4.

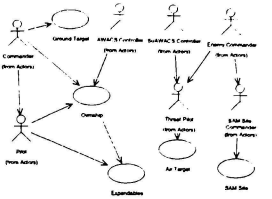


Figure 4 Use-Case View of Mission

The Interaction and Collaboration Diagrams (Scenario) show how the objects, defined in the Use Case analysis, interact as part of a scenario, as shown for a small segment of the mission on Figure 5. The Scenario Diagram is also used to determine the classes and additional objects that are needed and their relationships with each other.

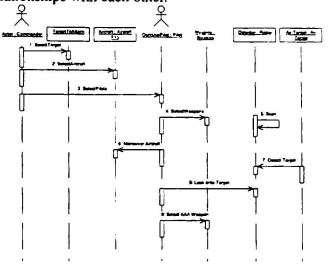


Figure 5 Scenario Diagram

Classes are graphically added to the Class Diagram and used to define abstract "real world" objects. The Class Diagram also shows the interaction between the classes including inheritance and associations as shown on Figure 6.

In this case, the Missiles, Bombs, and Guns are "a kind of" weapon and inherit from the Weapons class. The Avionics, Airframe, and Weapons are "part of", or aggregates of the Ownship. There is also an association between the Target, Ownship, and SAM_Site, which allow messages to be sent between them. The figure also shows which packages the classes belong to that provides a way to group similar classes and objects.

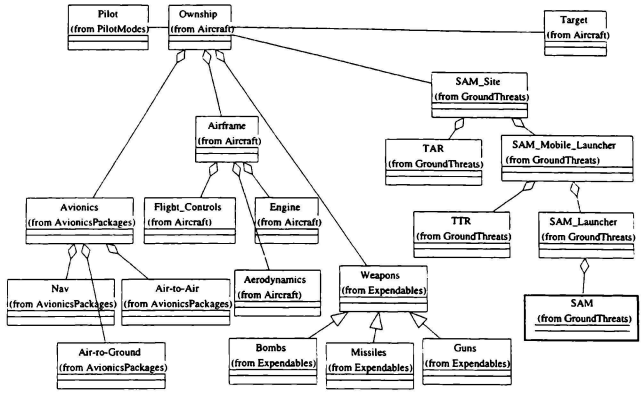


Figure 6 Class Diagram

The State Diagram is used to graphically generate the state of the system. Since Rose allows the user to switch between the different views of the problem, the user may adapt the sequence to his/her own preference. The State Diagram also is used to allocate classes and objects to modules in physical systems, show dependencies, and compilation order. An example of a State Diagram is shown on Figure 7.

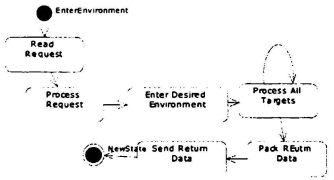


Figure 7 State Diagram

Lastly, the Deployment View is often used to assign tasks to processors, map a distributed network, or other communications features. It shows the configuration of runtime processing elements.

Common Modeling Architectures

DIME, GTSIMS-DISAMS[21] and MOSAIC are some common government architectures. All take advantage of modern computing to an extent to achieve the merging of multiple complex model instantiations.

DIME

DIME uses the concepts of cradles to make legacy codes in its tool kit (e.g., ESAMS) communicate through a single GUI interface allowing the user to set-up, run the individual models against a common problem, each solving its particular portion. Each model is run separately, but the assumptions and inputs used plus outputs generated relate to the large end-to-end engagement situation, oft times feeding information to the other models' independent runs through a series of data translations. DIME support the user's analytical development with

- common GUI for situation input, with graphical and raw data presentations, written in FORTRAN, 'C' using X11/MOTIF
- input data bases for each of the cradled sub-models with some values automatically filled in for user
- background (ordered queue) processing of the necessary conditions for the various sub-models
- graphical and text output post-processing and playback through FORTRAN and 'C' codes in combination with X11/MOTIF
- Support for SGI, Sun workstations
- Users and technical manuals

GTSIMS-DISAMS

GTSIMS-DISAMS grew from the need to routinely place the detailed missile models into a complex, detailed IR/EO scenic environment: aircraft motion, aircraft signature propagation, warning systems, countermeasure, and terrain effects. The GTSIMS portions handled the creation of the requisite input databases and hook together for DISAMS execution. GTSIMS-DISAMS supports user's analytical development with:

- graphical menu selection input option, which uses 'C' driven X11 and TCL, with select supporting graphics
- a data base of preset environmental conditions, select aircraft, some countermeasures, and specific IR/EO SAMS for mix-and-match situation creation
- background and batch processing on individual and heterogeneous network
- graphical and text output post-processing analysis and playback through 'C' and FORTRAN codes in combination with UNIX freeware like XV plus X11, TCL
- Support for Alpha, Sun, and SGI workstations,
- An extensively verified/validated set of missile and environmental models, and
- User manuals, technical references, user support, plus training classes.

Recent upgrades to GTSIMS-DISAMS have restructured the code into a more object-oriented design.

MOSAIC

MOSAIC also grew from the need to hook together legacy models together with new capabilities for simulating IR missile engagement and countermeasure situation. MOSAIC supports creating the engagement and post-processing of results with

- graphical menu selection input environment, written in C++ and utilizing X11/MOTIF and GKS, with select supporting graphics,
- a data base of preset aircraft, dispenser systems, countermeasures, and specific IR/EO AAMS, SAMS,
- pre-run AAM/SAM fidelity selection via model choice (DISAMS, TRAP, TEAM),
- foreground and batch processing on individual and networked SUN workstations,
- graphical post-processing analysis and playback though C++ and FORTRAN driven calls to GKS, Khoros, XGRAPH, XV,
- verified/validated models of air vehicles, missiles, countermeasures, and environmental models, and
- User Manual plus user support.

Dynamic Level of Detail Control

Users may want to dynamically switch model entity levels of detail during a run. This is possible with JMASS by compiling and building teams of players that all contain the levels of detail needed using one of several methodologies. Previous papers discussed downgrading the individual players to assemblies and adding a new player broker which interacts with the user/simulation which can dynamically select the level of detail desired. This was done using a port to the player broker that receives inputs on the version of the player to use and calls the appropriate assembly. A multi-level target, capable of running at varying levels of fidelity, was developed using JMASS in 1996.[22]

Even in constructive usage models, JMASS or similar flexible structures, can provide the user with the appropriate level of fidelity, as required, for the problem being analyzed. Continuing with an IR/EO engagement analysis example, the guided missile engagement simulation varies the amount of fidelity required with respect to the missile-to-target aircraft ranges involved. In a sufficiently long-range engagement, the target's extent is little more than a "dot" or pixel to the viewing seeker. At this point, a simplified model of the target, in correct context to its environment, is sufficient for establishing lock-on and initial track. A set of Analytical Model representation of target and missile suffice. As the range closes, the target becomes a shape with discernible characteristics. Here the target model needs more detail -- e.g., body shape/heats, principal reflection, plumes -- and the missile seeker requires more sophisticated tracking/guidance, CCM logic. Dynamic models of the associated phenomena and logic would work at this point. Once the engagement reaches end-game situation, a very high level of detail is required for the models of both the target and missile. This is an EW environment situation, and emulative models are the correct choice.

For the same situation, legacy model developers made assumptions as to the necessary level of detail needed for their models' anticipated usage. As a result, EW capable (countermeasure/counter-countermeasure

models) tend to work the entire problem at extreme detail at all times, even when it is not needed.

Interoperability Requirements and Applications

While it might be possible to evaluate engagements at an abstract, analytic level, model designers seeking to evaluate new communications channels ability to reduce or eliminate currently observed interoperability problems, have determined that such abstract models are not sufficiently detailed. Real-world experiments, like the All Services Combat Identification Evaluation Team (ASCIET)[23], have shown that communications models must evaluate the transmission, reception and integration of individual messages by the pilot or weapon system.

At the same time, even though the models must be very detailed, they must also be executed at a near real-time rate, or else they will not allow the model designer to evaluate the confusion or clarity that results from introduction of modeled capabilities into the field. In the Integrated Air and Missile Defense (IAMD) domain, the ability to exchange information with coordinating platforms and assimilate that information must far exceed the speed at which the engagement takes place. That way the pilot and/or weapon system can have sufficient time to develop situation awareness and select the proper response options.

The keys to evaluating the complex interactions that must take place in the IAMD domain is to effectively and efficiently sort threat objects from friendly, neutral, and unknown objects is:

(1) to collect all the information observed by organic sensors on ownship and exchange it via communications channels with other in-theater platforms (e.g., E-2C, AWACS, AEGIS ships, Patriot batteries) and

(2) to correlate and integrate all such information using identical or similar software so that each combatant has a clear and unambiguous combat picture.

Exercises, such as ASCIET, have shown that if in-theater systems fail to share all sensor information or if they correlate and integrate that information using different software, there will be a large number of threat systems completing their assigned missions and there will be some non-threat systems mistakenly shot down. This is why there is so much concentration on analyzing the interactions of Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) capabilities of current and proposed systems.

Industry and government are currently creating models to evaluate the impact new or modified systems have on the ability to develop a Single Integrated Air Picture (SIAP) for the entire theater. For the most part, the approach taken is to create an emulative model of the new or modified capability, then determine how to simplify the model to retain its essential design characteristics while achieving near real-time execution rates. The tools used to develop such models are drawn from the same set as those used to develop other communications channels, such as telephone exchanges or cellular networks.

One such tool is Rose RT and its direct ancestor, ObjecTime, which are focused on developing embedded real-time systems. ObjecTime, developed by a company of the same name, was originally conceived by Nortel engineers to design telephone exchanges / networks and evaluate their ability to handle surge events. When it became apparent that the emerging tool had the capability to evaluate system capabilities for a broader spectrum of applications, ObjecTime was spun-off from Nortel to market the tool.

The advantage of using Rose RT rather than ObjecTime is that it has transitioned to the Universal Modeling Language (UML) paradigm discussed previously for Rose, added a reverse engineering capability and automatic code generation into additional languages, and added support for additional host development platforms. Like ObjecTime, Rose RT also has an ability to characterize model behavior using hierarchical Statecharts, which are embedded in the model's class or component diagrams.

Development Cycle Support

It is important to use a spiral design methodology, so the modeling and simulation components that support system development and testing activities and capabilities can be reused during development and testing, as shown on Figure 7.

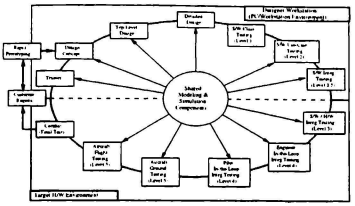


Figure 7 Testing Cycle

The development cycle begins when the customer identifies new requirements that the weapon system must satisfy. The contractor, to better understand the customer requirements, rapidly develops prototype concepts intended to satisfy stated customer requirements and tests them in operational scenarios endorsed by the customer to identify preferred design concepts that satisfy stated requirements as well as requirements revealed during the rapid prototyping process.

Once stated and derived customer requirements are well understood and preferred prototype design concepts tested, responsibility for developing each design concept is assigned to an Integrated Product Development (IPD) team, which is composed of representatives from several development and testing disciplines. The IPD team proceeds from prototype design concept to detailed design in a Software Development Facility (SDF), which can now reside in a PC/Workstation configuration.

At this point, a transition is made to test on the target hardware. Traditionally, software / hardware integration testing of the Mission Computer subsystem is accomplished in a Software Test Facility (STF). To verify that functionality has not been lost in the transition from host to target hardware, some of the testing accomplished on the PC/Workstation configuration can be repeated using the same tools. One advantage of using tools like Rose RT or Rhapsody is that they can be used in both the PC/Workstation and target hardware environment, so the IPD team need not transition to other tools. The next step is to test how well subsystems work together. Traditionally, this has been accomplished in two separate facilities. If the evaluation was done from an engineering perspective, then such testing was performed in the System Integration Facility (SIF). But if the evaluation was done from a pilot's perspective, then that testing was performed in a Manned Flight Hardware Simulator (MFHS). The distinction between these two testing environments is now blurring. Only when the pilot wishes to evaluate how the avionics system provides feedback on aircraft flying qualities is it essential to perform tests in the MFHS. Tools to support testing in these environments are similar to those used to collect flight test data. In addition, the missions / scenarios from previous phases can be reused using a tool like STAGE.

On-aircraft ground testing and flight testing is conducted to ensure that customer requirements have been satisfied in the production aircraft / weapon system and that the aircraft is capable of performing all assigned missions when flown by operational test pilots. Flight instrumentation systems monitor the interaction between subsystems as the pilot performs tests that are representative of various combat phases, and other airborne and ground-based monitoring systems establish actual performance of the aircraft / weapon system under test.

Trainers are designed to closely replicate how the aircraft / weapon system performs assigned missions, but now testing tools are designed to measure the performance of the trainee rather than the aircraft / weapon system. The trainer itself very much resembles a high fidelity Development Simulator and shares many of the same modeling and simulation components used in that environment.

It is essential that models used during different phases of the development cycle belong to the same family, so that the software in development is always evaluated in, what is essentially, the same operational environment, regardless of development or testing phase.

Conclusions

The use of graphical tools to generate new and updated models has proven to be very effective, if the developer designs the models with reuse and common interfaces in mind. It is important to store the design, as well as the code, for the models in a reuse library. The graphical design tools should contain a reverse engineering capability to allow legacy models to be updated / converted to the standard format and allow changes to be made to the models at the model design level and automatically added back into the code.

Existing models should use wrappers to allow the models to be used. This allows existing model to take advantage of "real world" interfaces, and allows a "real world" hardware and software models to be substituted. The start-up costs vs. the long-term maintainability cost

of using legacy models should be addressed early. Even though setting-up a wrapper for an existing model is often faster and easier, maintaining the model over the long term could be much higher.

To maximize M&S component reuse, components should be designed in a modular fashion, with well-defined interfaces. Design logic/rationale of shared components should be maintained in a reuse library to facilitate valid reuse decisions and contain:

- object-based designs that support component reuse,
- a repository of models and their components, and
- documentation/testing.

Future Applications

Legacy Models

The models for a simulation can be converted in two ways:

1. Using wrappers on existing legacy models with common interfaces and
2. Using new models at different "levels of detail" based on reusing the functional behavior code of the legacy models with object-oriented tools and processes.

A comparison of the execution and development time using the two methodologies will provide lessons learned, as well as working models, for a simulation. Generating interfaces for different "level of detail" models will provide a baseline of how to change from one modeling level to another during execution.

Avionics Design and Testing

The architecture used for object-oriented models can also be used to support building embedded software for flight. Many of the same design principles apply as well as the desire to be able to develop on one platform and then move the code to different target hardware. In addition, it would be useful for the software developed for a simulation to be the same as that used in flight. There are several additional features that are needed for simulation software, like initialization in flight, slewing to different areas of the world, stopping, and starting the simulation, that is not normally included in flight software. With the object-oriented principles and the use of polymorphism, a flight hardware load could be very similar for the development, simulation, trainer, and flight hardware version. If successful, it would have a significant impact on overall program affordability.

---

[1] TAC Thunder Overview Briefing, 1995.
[2] US Army Space and Strategic Defense Command, EADSIM, 1995.
[3] TAC Brawler User Manual, 1994.
[4] Suppressor User Manual, 1993.
[5] Digital Infrared Seeker and Missile Simulation, Users Manual, February 199.
6 Enhanced Surface-to-Air Missile Simulation (ESAMS) Technical Capabilities Sheet, 1995.
[7] Gen. Case Briefing, AFXOM, 1994.
[8] Digital Integrated Modeling Environment, User Manual, 1998.
[9] Modeling System for the Advanced Investigation of Countermeasure, User Manual, 1996.
[10] JWARS Briefing, May 1996.
[11] TRD 041896, Joint Simulation System (JSIMS) Joint Program Office (JPO), "Technical Requirements Document", 4/19/96 (Draft).
[12] JMASS ORD, March 1996 (Draft).
[13] Beavin, W., Emrich, T. and Bezdek, W., "DIS Interface Using JMASS", NAECON, May 1995.
[14] HLA Briefing, Dr. Judith Dahlman, Dec. 1994.
[15] Georgia Tech Simulation Integrated Modeling System - Digital Infrared Surface-to-Air Missile Simulations, Users Manual, April 1997.
[16] Integrated Systems, Inc., System Build User's Guide, 1996.
[17] Virtual Prototype STAGE, 2000.
[18] UML for Rational Rose 4.0, Rational Software Corporation, 1996.
[19] Toni Malins, Rhapsody Object-Oriented Analysis, Design and Implementation Tool, I-Logix, Inc., 1997.
[20] Unified Modeling Language (UML), Rational Corporation, 1996.
[21] Georgia Tech Simulation System - Digital Infrared Surface-to-Air Missile Simulations, Users Manual, 1997.
[22] W.J. Bezdek & E.A. Reidelberger, "Vehicle Interactive Digital Pilot using JMASS", AIAA 96-3494, July 1996.
[23] C4I-Pro-Digest, Vol 2 #519, October 31 1996.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37274

AIAA-2000-4401

AN HLA BASED FLIGHT SIMULATION ARCHITECTURE

Wim Huiskamp
Hans Jense
Henk Janssen
TNO Physics and Electronics Laboratory
Command & Control and Simulation
P.O. Box 96864
2509 JG The Hague, The Netherlands
Huiskamp@fel.tno.nl

Keywords:
Simulator Architecture, Simulator Components, middleware, HLA-RTI

ABSTRACT: *Modern distributed simulation concepts such as the US DoD High Level Architecture (HLA) support interoperability between heterogeneous simulations, thus enabling the development of flexible, re-usable simulation Federations. Based on the principles of HLA, the Dutch national collaborative project 'SIMULTAAN', has extended this concept to the simulation component level. An individual simulation, participating in a Federation, can itself be composed of various Components interacting through the same architecture and underlying infrastructure as the overall Federation. Thus, Federates participating in a Federation can be viewed as a Federation of Components. The Federation of Components is represented to the Federation of Federates by a specific Component known as Federate Manager. This 'gateway' provides protection of (restricted) Component data-traffic and also reduces wide area network bandwidth requirements.*

This paper discusses the SIMULTAAN Simulator Architecture (SSA) and its application in the development of a component based Flight Simulator. SSA Federates communicate via a data-exchange middle-ware layer, called the Run-time Communication Infrastructure (RCI). The RCI is currently based on the HLA Run-Time Interface (RTI), but allows the use of other standards as well. The innovative approach of the SSA is that the RCI extends the Federate interoperability concepts of HLA by providing data-exchange between SSA Components in a similar way. With this approach the RCI abstracts components from the intra-SSA Federate protocol and network hardware.

1. Introduction

A (flight) simulator is usually constructed from several hardware and software components, which may be manufactured by different parties. Component technology can be used for rapid and cost-effective development of simulators through re-use and exchange of existing simulator components.

Components are considered the basic building blocks of a simulator, which can potentially be used for more than one type of simulator. Examples of simulator components are a flight dynamics component, a component that visualizes the Out-The-Window virtual environment, a component that handles the Human-Machine-Interface (HMI) or a motion platform component. Consequently, a simulator can be thought of as a set of interacting components. The total functionality of the simulator may be expressed as the 'sum' of the functionality of its constructing components.

In order to apply component technology successfully, the component/simulator architecture has to fulfil certain conditions: the functionality of each component must be well defined and the interfaces between components have to be defined in a formal manner. Formally described interfaces reduce incompatibility problems that might otherwise not be noticed until the integration phase. A clear distinction between the interface(-description) of a component and its functionality improves re-use of that component since the interface agreements of a new simulator usually imply modifications of the interfaces of available components. The architecture must also support a mechanism to coordinate the overall behavior of the components, which should for example ensure proper initialization of components before they are allowed to exchange simulation data.

Copyright © 2000 TNO-FEL. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

This paper describes the SIMULTAAN Simulator Architecture (SSA) [4] which was developed in the framework of the Dutch SIMULTAAN project and provides an example of the uses of SSA in a Flight Simulator application.

The paper is organized as follows. Section 2 introduces the general concepts of HLA, section 3 describes the Simultaan Simulator Architecture (SSA) Section 4 details the development of a typical Flight Simulation Federate based on a combination of COTS and in-house Components. Finally, conclusions are drawn in Section 5.

2. High Level Architecture HLA

The US DoD has defined the High Level Architecture (HLA) as the standard technical architecture for all DoD simulations. The HLA development is targeted at two main objectives:

1. To achieve interoperability amongst heterogeneous simulations, e.g., real-time virtual simulations and non real-time constructive and live instrumented entities;
2. To promote re-use of simulations and their components by specifying the general structure of the interfaces between simulations without making specific demands on the implementation of each simulation.

In the HLA baseline documents [1], three basic concepts are defined:

- The Interface Specification is a formal, functional description of the interface between the HLA application and the Run-Time Infrastructure;
- A set of technical rules is defined to which an HLA participant has to comply;
- The Object Model Templates are standardized formats to define the functionality of simulation models and the interaction between models [3].

The Run-Time Infrastructure (RTI) is an implementation of the formal Interface Specification of HLA. It implements a distributed operating system and forms the basic software layer for HLA applications.

An HLA *Federation* contains a number of members called *Federates*. Federates may be simulation models, data collectors, simulators, autonomous agents or passive viewers. A simulation session, in which a number of Federates participate, is called a *Federation Execution*. Simulated entities, such as vehicles or aircraft, are referred to as *objects*. Live entities can be mapped onto objects (by using simulation surrogates), so that they can appear identical to simulated entities. All possible interactions between the federates of a federation are defined in a so-called *Federation Object Model* (FOM). The capabilities of a federate are defined in a so-called *Simulation Object Model* (SOM). The SOM is introduced to encourage reuse of simulation models.

The state of each object is defined by its *attributes*. Attribute values can be passed from one object to another. Objects interact with each other and with the environment via *interactions* which may be viewed as unique events, such as a collision between objects. Initially, an attribute is controlled by the federate that instantiated the object whose state the attribute is part of. However, attribute ownership may change during the execution. This mechanism allows, for instance, the fidelity of a vehicle simulation to be adjusted by passing its position attribute from one kinematics simulation to another.

In order to reduce network traffic and limit the amount of computation each federate has to perform, the HLA provides a mechanism of *Publication* and *Subscription*. Upon initialization of a federation execution, each simulation registers (with the RTI) which objects and which attributes it will represent (publication). It also registers which attributes and interactions it needs in order to be able to perform its task (subscription). Note that a federate can not only subscribe to attribute types, but also to (ranges of) attribute values. The aim here is to filter as much information as possible at the source. Both publications and subscriptions are dynamic and may be changed during a session. In this way, multicast communication is realized which reduces the necessary bandwidth.

The Run-Time Infrastructure supports HLA compliant simulations through a number of services. The main categories of services are:

- Federation Management: create/destroy execution (of a federation), join/resign execution (of a federate), pause/resume execution;
- Declaration Management: subscription/publication;
- Object Management: instantiate/delete (an object), update/reflect (an attribute), request/provide (an attribute), send/receive (an interaction);
- Ownership Management: transfer ownership (of an attribute) from one object to another;

- Time Management: handling of messages in different ways, depending on the requirements of their destinations, e.g., in receive order (appropriate for real-time human-in-the-loop simulators), in priority order, in causal order, or in timestamp order. These last three methods are typically suitable for non real-time event driven simulations for, e.g., analysis purposes.

For details of the HLA interface specification, see [1]. Documents describing HLA can be downloaded via the Defense Modeling and Simulation Office (DMSO) homepage (http://www.dmso.mil).
HLA development is supported by a well defined and structured process known as Federation Development and Execution Process (FEDEP) [2].

3. SIMULTAAN Simulator Architecture

SIMULTAAN was a 2.5 years project which brought together knowledge and experience in the area of simulators and distributed simulation from universities, research institutes and industry in The Netherlands. Two main results of the project can be distinguished:

1. *SIMULTAAN Simulator Architecture (SSA).*
   A generic framework applicable for a wide range of simulators.
2. *Permanent Intellectual Infrastructure.*
   The SIMULTAAN consortium strengthened working relationships between its partners.

The SSA defines a simulator Component architecture that addresses the need for an efficient federate development process and makes effective use of simulator component technology. The SSA is intended to maximize the re-use potential of components by defining a standard interface for simulator components. In this way simulator development time will be reduced. By requiring that components comply to the standard interface and comply to a number of rules, re-use is possible in other simulators built on the same architecture.

On the level of Federates and Federations, the SSA is fully compatible with HLA. As an extension to HLA, the SSA introduces a new level, that of the Federate Component (See Figure 1). The SSA facilitates interoperability between Components inside a Federate, in a similar manner as HLA-RTI does between Federates.

The SSA identifies the following key architectural elements: *Component*, *Run-time Communication Infrastructure (RCI)*, *Federate Manager (FM)*, and *Federation Manager (FedMan)*.

A *Component* is the basic building block for an SSA Federate. All Components interact with the simulation environment through a standard interface, provided by the Run-time Communication Infrastructure.

The *Run-time Communication Infrastructure* (RCI) is an object-oriented middle-ware layer for exchanging data between Components as well as between Federates. The RCI provides an abstraction layer to shield the Component developer as much as possible from the underlying interoperability standards.

Each SSA Federate is built from a set of Components with one obligatory Component, called the *Federate Manager*. The Federate Manager acts as an intermediary between the Components in the Federate and the rest of the Federation; it represents the Federate to the Federation. This 'gateway' provides protection of (restricted) Component data-traffic and also reduces wide area network bandwidth requirements. Note that bandwidth- and latency requirements for data-traffic between Components are typically much higher than for traffic between Federates. This fact is a consequence of the finer granularity of Components.
The Federate Manager requires a special version of the RCI that is equipped with two Communication Servers: one for local communication within the SSA Federate, and one for global communication between the SSA Federates.
The Federate Manager keeps track of the state of its Federate and its Components and responds to commands from the Federation Manager.

The SSA *Federation Manager* is an SSA Federate that controls the behavior of the Federates within the Federation by issuing commands to the Federate Managers (like start execution, stop execution, pause execution).

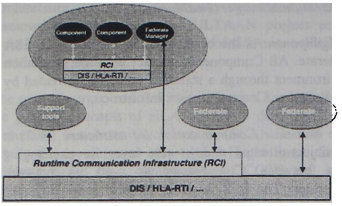


Figure 1 *HLA based Component architecture*

Recapitulating the above: each SSA Federate is implemented as an HLA federation with its Components acting as federates. This results in one overall SSA federation and additional federations for each participating SSA Federate. The overall SSA Federation consists of all Federate Manager Components (representing the participating SSA federates) plus the Federation Manager.

In addition to the identified elements in the architecture, the SSA consists of the SSA Rules, the SSA Interface Specification and the SSA Object Model Templates.

The SSA Rules define the responsibilities and relationships in an SSA federation. The most prominent SSA Rule is that all Components must adhere to the SSA State Transition Diagram (SSA-STD), which is depicted in Figure 2. The SSA-STD is used by the Federate Manager to coordinate the state transitions of the Federate during execution. The Federate Manager prepares its Federate for joining the Federation and when the Federate has joined the Federation, the Federate Manager initiates and checks the state transition of the Components in its Federate, as requested by the Federation Manager.

The SSA Interface Specification (SSA-IF) is a formal, functional description of the interface between the application and the Run-time Communication Infrastructure (RCI).

The SSA Object Model Templates (SSA-OMT) are standardized formats to define the capabilities of Federates and Components and their respective interactions. The SSA-OMT is equivalent to the HLA-OMT [3]. The different object models used in the SSA are presented below.

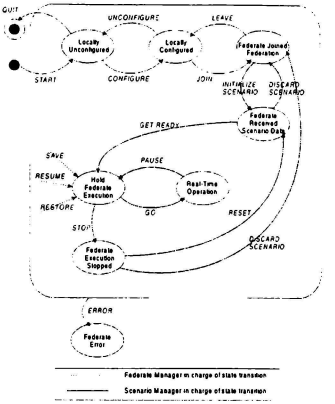


Figure 2 State Transition Diagram

Federate Development

Federate Development describes the way developers produce federates. New Components may have to be developed or existing Components may need to be adapted. During the design process, such needs will be identified and translated to Component requirements. The Federate Development process results in a validated simulator or tool. A Federation is created by producing its Federates and defining the interactions between them in an HLA Federation Object Model (FOM).

Requirements for the Federate are specified in cooperation with the end-user and can be regarded as a starting point for the development. From the user requirements, the system requirements are identified which initiate the design process of the SSA Federate.

On the Federation level, the SSA identifies the SSA-SOM and the SSA-FOM, both equivalent to the HLA-SOM and HLA-FOM. On the Component level, the SSA identifies the SSA-COM and the SSA-SCOM, which are explained next.

Each SSA Component has a Component Object Model (SSA-COM). This object model formally specifies the attributes and interactions a Component publishes to

other components. It also specifies the attributes and interactions a Component will subscribe to during run-time.

Each SSA Federate is build from a set of interoperable Components. The interactions and attributes that are exchanged between all Components of one Federate, including the data that is exchanged with other SSA Federates, is formally described in the SSA Simulator Component Object Model (SSA-SCOM). The difference between the SCOM and the SOM is that the latter merely describes the interface between SSA Federates and not the intra-Federate communication between the Components.

The SSA-COM and the SSA-SCOM object models have similar roles on the component level as the SOM and the FOM have on the federation level. Both the SSA-COM and SSA-SCOM descriptions are expressed in the HLA-OMT format.

Distinction between COM and SCOM on one hand, and SOM and FOM on the other hand, enables different treatment of local and global communication. Local communication is the exchange of information between Components in a Federate, whereas global communication is the exchange of information between Federates. Distinguishing between local and global communication allows for a mapping of the local communication onto a dedicated network that is able to handle intra-Federate high-speed data exchange while having the possibility to communicate with the outside world at another data exchange rate and possibly via other physical network media.

The SSA object models enable clear specifications for the capabilities of Federates and Components. Federate and Federation development in SSA is analogous to the HLA Federation Development and Execution Process (FEDEP) [2].

An SSA Federate will be designed with optimal use of existing components. Therefore access is needed to the descriptions of object models and components that are available in the SSA Object Repository (SOR). The SOR will contain SSA compliant simulators, components and tools. It may also contain configuration, initialization, and validation data. The SOR allows controlled access by the SSA partners.

Run-time Communication Infrastructure

The SSA is the common high-level architecture for Component based simulators and tools. The SSA provides services to both the Components and the Federate, which is the distributed composition of the Components. All Components interact with the simulation environment through a standard interface provided by the Run-time Communication Infrastructure (RCI). The RCI is the implementation of the SSA Interface Specification.

The RCI provides the component developer with the necessary functionality to incorporate the Component into an SSA Federate. The RCI provides a protocol-independent interface to the simulated environment. The design of the abstraction layer and the Application Programmer's Interface (API) have been discussed in detail in a previous paper [4], and is briefly summarized below.

The RCI consists of two separate software layers, one is called the *Environment* and the other is called the *Communication Server* (see Figure 3).

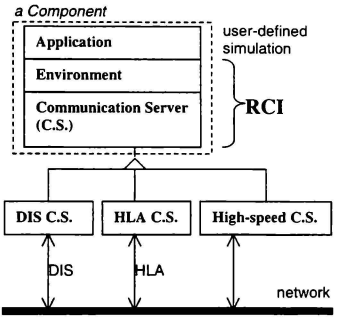


Figure 3 RCI structure

The Environment provides Components with an overview of both the Federate and the Federation. The Environment reflects the current state of the Federate, i.e., the state of all its components. It allows the addition and deletion of components. Furthermore, the RCI allows components to subscribe to relevant information. Each Component can publish data to which other Components of the Federate may subscribe. Components can send and receive events. The translation of the events to a specific interoperability standard (such as DIS or HLA) is left to the Communication Server.

The Communication Server represents the layer that takes care of the actual communication. Its function can be compared to that of the HLA-RTI. In a way it represents a distributed operating system. The interface between the Environment and the Communication Server is based on the HLA Interface Specification [1]. A Communication Server communicates with other Communication Servers to exchange object and event information to keep the Environment up-to-date. Currently, the Communication Server is based on the HLA RTI, but other standards are possible. Dedicated versions of the Communication Server can be implemented for the support of specific simulation protocols or network layers. This requires only minimal changes in the application-specific source code as it merely interacts with the Environment.

The innovative approach of SSA is that the RCI extends the Federate interoperability concepts of HLA by providing data-exchange between SSA Components in a similar way. Components use equivalents of the HLA federation management services, declaration management services, object management services etc. In this way, the RCI abstracts Components from the intra-SSA Federate protocol and network hardware, and it establishes a clear separation between communication aspects and application-specific aspects. This enables a Component developer to focus on the required functionality of the specific Component rather than the technical details of the communication aspects.

To further facilitate the developer with an abstraction of the communication it is noted that the simulation objects and the simulation events are *formally* described through its Component Object Model (SSA-COM). This enables the use of automatic code generators to construct object-oriented classes (for instance C++ or Java) for each simulation object and simulation event in the COM. The automatic code generation approach has proven to be highly successful in SIMULTAAN. The generated code shields the application programmer from doing elaborate bookkeeping concerning attribute updates, while making use of the encoding and decoding facilities offered by the RCI to communicate attribute and parameter values along the physical network.

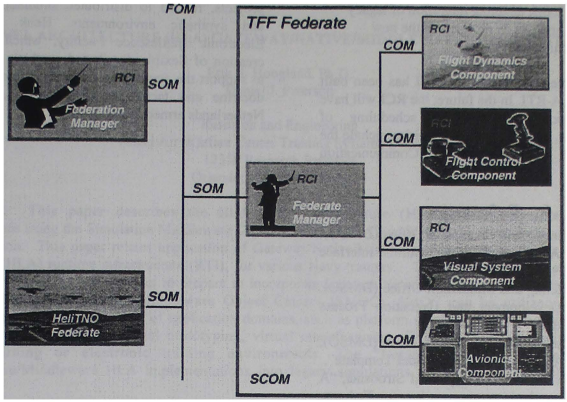
4. Implementation and Demonstration

A functional proof of concept of the SSA architecture has been presented in a large demo on the 24th of June, 1999 at TNO-FEL in which all SIMULTAAN partners participated. A further demonstration of SSA was developed by TNO-FEL and demonstrated at ITEC2000 in The Hague. The *TNO Fighter Federate (TFF)* developed by TNO-FEL shows the Component approach for a generic fighter simulator. The *TFF* consists of a number of Components using the RCI as interoperability layer. Some Components are based on Commercial Off-The-Shelf (COTS) software linked to the RCI (e.g. VPI's FLSIM), while others are in-house TNO developments (e.g. Out-The-Window Visual). A scenario was shown consisting of a manned fighter aircraft and computer generated targets (helicopters).

The *TFF* Federation is mapped onto an HLA Federation. This approach is straightforward as the SSA-SOM and SSA-FOM are identical to the HLA-SOM and HLA-FOM.

The demonstrated Federate is constructed from the following Components (see figure below) :

- *Aircraft Dynamics Component*,
  The *TFF* Flight-Dynamics Model Component, based on Virtual Prototypes (VPI) re-configurable flight simulator (FLSIM) software, provides a flexible, 6 DOF flight model for fixed wing aircraft. FLSIM and the RCI execute as separate 'threads' linked through shared memory.
- *Flight controls and cockpit mock-up Component*.
  Basic flight controls (stick, throttle and rudder) communicate through the RCI with the Flight Dynamics component
- *Avionics Component*.
  The kernel of this Component is based on VPI's 'Virtual Application Prototyping System' (VAPS) software. Vital instrument settings (e.g. radar display) are linked through the RCI with attributes simulated by other *TFF* Components.
- *AA missile simulation Component*,
  Simplified model or missile sensors system and flight dynamics behavior.
- *Visual System Component (OTW display)*
  The Visual System Component, a development of TNO-FEL, is based on SGI Performer and is linked through the RCI with other components. The visual scene rendered for the *TFF* is the 'Wales' database developed by Aerobel UK.



- *Federate Manager*
  The Federate Manager presents the federation of components to the federation.

In the figure above a schematic view is presented of the Federates that participated in the ITEC2000 demonstration. Each of the previously described Components executed on a separate computer. The computer hardware consisted of a mixture of Unix based machines (e.g. SGI Onyx for OTW visual, SGI O2's for Dynamics model and Avionics) and Linux machines (Federate Manager). A medium fidelity mock-up with basic controls (stick, rudder and throttle) was provided.

TFF was designed and built within 2 months. The RCI middle-ware and companion Code-generator tools allowed rapid conversion of existing legacy simulation software into an SSA compliant Component : development effort for the FLSIM conversion was about one week.

5. Conclusions

This paper discussed the SSA architecture for simulator development. The architecture is intended to maximize the re-use potential of Components by defining a standard interface. Components that comply to the standard interface, and comply to a number of rules, can be re-used in another simulator built on the same architecture.

The global design strategy for the SSA was presented, followed by an example application. The *TNO Fighter Federate* demonstrates the architecture's capability for developing flexible, re-configurable, interoperable simulations in an easy to use, powerful development environment. Although the most direct application is the development of training simulators, other application areas such as procurement/acquisition, weapon system (cost) effectiveness studies and simulator research can also benefit from the SSA concept.

In addition to full compliance with HLA, the innovative architectural concept supports the key capabilities required for future simulation applications :

- The use of simulation Component technology, supported by a set of development tools, to facilitate design-for-reuse and ease system integration;
- An abstraction layer (or middle-ware) and a code generator hiding complexities of the underlying interoperability standards and enabling simulation protocol migration with minimal changes to the functional implementation of the Component.

- A structured development process, supported by appropriate tools, enabling migration of legacy simulations and COTS products to the new standard architecture;

The current implementation of the RCI has been built on top of the HLA-RTI. In the future, the RCI will have additional support for real-time scheduling of Component tasks. Further planned activities include the development of a high-speed RCI Communication Server.

6. References

[1] Defense Modeling and Simulation Office (DMSO), "High Level Architecture for Simulations Interface Specification".

[2] Defense Modeling and Simulation Office (DMSO), "Federation Development and Execution Process (FEDEP) Model".

[3] Defense Modeling and Simulation Office (DMSO), "High Level Architecture Object Model Template".

[4] Marco Brasse, Wim Huiskamp, Olaf Stroosma, "A Component Architecture for Federate Development", Simulation Interoperability Workshop, Fall 1999

Author Biographies

Wim Huiskamp is a research scientist in the same division. He holds an M.Sc. degree in Electrical Engineering from Twente University of Technology, The Netherlands. He works in the field of High Performance Computing and Networking and specializes in design and implementation of distributed computer architectures. His research interests include system architectures, real-time visual simulation and multi-media technology. Wim was the TNO-FEL project lead for SIMULTAAN.

Hans Jense is a senior scientist in the Command & Control and Simulation Division at TNO-FEL, where he currently heads the Simulators Group. He has been the technical lead in various projects related to distributed simulation technology and virtual environments. He holds a Ph.D. in Computer Science from the University of Leiden.

Henk Janssen is a project manager in the Simulators group of TNO-FEL. He holds an M.Sc. degree in Aerospace Engineering from Delft University of Technology. Henk started his professional career in 1985 in the area of complex simulations of weapon systems. In his function as project manager is also most of the time involved in the technical aspects of his projects, related to distributed simulation technology and synthetic environments. Henk manages the Electronic Battlespace Facility, which enables the creation of flexible distributed synthetic environments to support the acquisition , training, system evaluation, doctrine and tactics development processes of the Netherlands armed forces.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37275

AIAA-2000-4402

HIGH LEVEL ARCHITECTURE (HLA) GATEWAY/NATIVE/MIDDLEWARE IMPLEMENTATIONS

Erik S. Hougland, Ph.D.
Daniel J. Paterson

Research and Engineering
Naval Air Warfare Center Training Systems Division
12350 Research Parkway
Orlando, Florida 32826-3275

ABSTRACT: This paper describes the High Level Architecture (HLA) Gateway/Native/Middleware implementations using the Simulation Middleware Object Classes (SMOC) code that was developed by NAWCTSD Orlando Florida. This paper relates application of Gateway/Native/Middleware architectures to the High Level Architecture (HLA) runtime infrastructure (RTI), for various Navy trainers. This paper characterizes knowledge areas that we re-investigated in order to prepare to incorporate Gateway/Native/Middleware modes into our HLA interface called the Simulation Middleware Object Classes (SMOC). SMOC can be utilized to facilitate interoperability across a diverse set of application domains, such as platform simulations, C4I, mission planning, simulation based acquisition, virtual prototyping, virtual manufacturing, financial models, cognitive models, distance learning or electronic training environments. The "Lessons Learned" in incorporating Gateway/Native/Middleware HLA implementations into legacy simulations using the SMOC architecture are presented.

INTRODUCTION

The High Level Architecture is one element of the overall Department of Defense (DOD) plan for the use of Modeling and Simulation to meet its mission that is being guided by the Defense Modeling and Simulation Office (DMSO).

As indicated in Figure 1 Modeling and Simulation Considerations, Advanced Modeling and Simulation is used to effect a cost effective and affordable solution to the problem of decreasing resources to support operations of all types and the increasing need for monetary and physical resources to test and train increasingly complex operations and technology. [1]

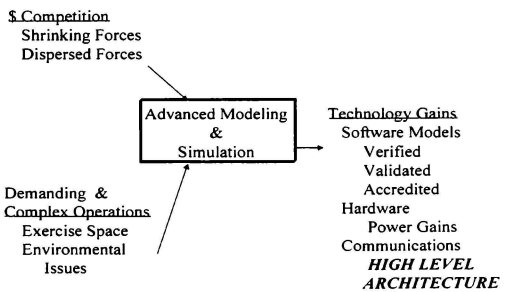


Figure 1 Modeling and Simulation Considerations

This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

It has always been an objective in the military forces to "Train as You Fight" with the actual forces that will participate in a mission. The modeling and simulation vision of the DOD will enable this objective in individual training, team training, mission rehearsal, mission planning, and even in acquisition. A potential global implementation of this vision is illustrated in Figure 2 Distributed Simulation Vision.

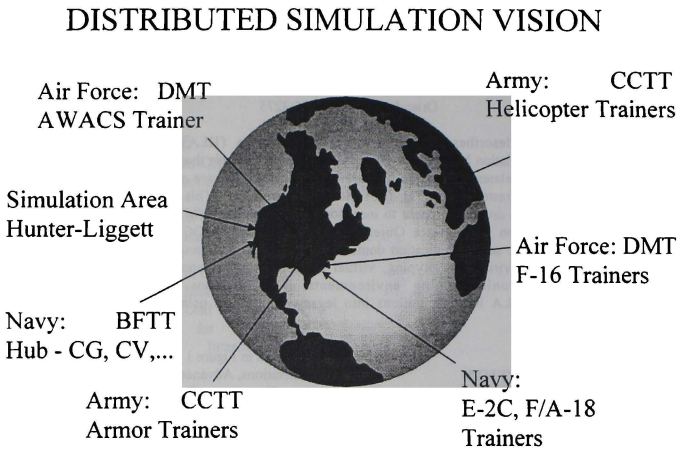


Figure 2 Distributed Simulation Vision

This is the vision of "Interoperability." The role of the Defense Modeling and Simulation Office (DMSO) is to bring this vision into being.

The DMS strategy to implement this vision is threefold:

1. DMSO has encouraged and supported the development of verified, validated, and accredited models of the mission space and its components.
2. DMSO monitors the development of hardware to ensure that modeling and simulation software makes the best use of hardware developments, and
3. DMSO supports the development of distributed networking software to link the simulations.

This strategy uses a network and distributed operating system software to connect the geographically distributed nodes hosting the various live simulations, constructive, or "Man-in-the-loop," simulations, and virtual, or fully model based, simulations. The software that supports this networking is the High Level Architecture (HLA) Run Time Infrastructure (RTI). It has evolved during the last decade as the end product of a standards development process. The current version of the DMSO RTI is 1.3 - "Next Generation" Version 3 (RTI-NG 1.3v3). It is available for free download from the DMSO HLA web site, http://hla.dmso.mil , after a free registration. Table 1 RTI - NG 1.3v3 Availability, lists the RTI software available by hardware platform, operating system, and C++ compiler. [1]

Table 1 RTI - NG 1.3v3 Availability [1]

PLATFORM / Binding	Operating System	C++ Compiler
Sun	Solaris 2.7	SPRO C++ 5.0
	Solaris 2.6	SPRO C++ 4.2
	Solaris 2.7	GCC 2.95.2
SGI	IRIX 6.5.6 N32 Mips 4	MIPSPRO C++ 7.3
	IRIX 6.5 N32 Mips 3	MIPSPRO C++ 7.2
Intel	Windows NT 4.0 SP 5	MSVC C++ 6.0 SP3
	Linux 2.2 (Red Hat 6.1) x86	Egcs 1.1.2
PowerPC, Motorola 680x0 SPARC, ARM MIPS (R3000, R4000, R4650) Intel (i960, x86, i386, i486)	Vxworks 5.3.1	Green Hills C++ 1.8.9
Java Language Binding	Sun Solaris 2.6	
	Linux 2.2 (Red hat 6.1) x86	Egcs 1.1.2
	Win NT 4.0	
CORBA Language binding	Sun Solaris 2.6	
ADA 95 Language Binding		

The requirements placed upon the HLA RTI are analogous to the requirements we place upon our languages to provide communications between people. The DMSO sponsored High Level Architecture (HLA) implements these requirements. The total DMSO package includes both the RTI and support software, or "tools." Table 2 HLA and Language Functions describes these requirements:

Table 2 HLA and Language Functions

FUNCTION	LANGUAGE	HLA
IDENTIFIED PEOPLE, PLACES, OR THINGS	NOUN	OBJECT
Descriptors of above	Adjectives, Prepositional Phases	Attributes
ACTION	VERB	INTERACTION
Descriptors of above	Adverbs, Prepositional Phrases	Parameters
Union, or Disunion	Conjunctions and negations	Management Services: ▪ Declaration, ▪ Time, ▪ Data Distribution, ▪ Object Management
Selection, or qualification services		Data Selection, or Filtering, by Declaration and Data Distribution Services
Transmission Medium for Communications	Verbal and Written Expression	Run Time Infrastructure (RTI) distributed across computer network(s)
Rules for Communication using above functions	Rules of Grammar, or Syntax	1. HLA Rules for Federation and Federate Behavior 2. Interface Specification for RTI and Simulations 3. Object Model Template

Background and History

Modeling and simulation has been a DOD analysis tool for many years. Until the late 1980's though, the M&S efforts were independent efforts. When linked to other simulations, there was no uniform methodology to pass information from one simulation to the next. Models of military entities, such as threat missiles, were developed, and then redeveloped for one simulation project after another. The development of formal verification, validation, or accreditation mechanisms has been a fairly recent occurrence. These systems will ensure, for example, that training system A's ATOLL air to air missile model will be the same as, or produce similar characteristics as the ATOLL model in training system B. This is part of the "Fair Fight" concept.

The development of the SIMNET system for linking tank trainers to teach team tactics is considered by many to be the first significant networked training system. This was quickly followed by Distributed Interactive Simulation (DIS). DIS evolved in a series of workshops where subject matter experts extended the SIMNET concept to all military operations. They did this by developing a series of Protocol Data Units (PDU) that allowed simulations to operate with each other by passing standardized data units around a network.

One basic concept of a DIS exercise was that all players were independent and were provided the entire "Ground Truth" *via* broadcast. Each simulation then determined if the information received was needed. If it was not, it was thrown away.

By the mid 1990's, the modeling and simulation community realized that the broadcast of all data, both entity and environment, to all players would overload the projected training and simulation networks with their larger numbers of entities. The concepts of servers to handle local generation of large amounts of data and selective transmission, *via* multi-cast, of updated data and processing requests evolved into the High Level Architecture as we know it today.

How it works, in theory

The collection of participants in an HLA distributed simulation is the Federation. Each unique participant, e.g. a computer or an entity simulation, is referred to as a Federate. Each Federate has an object model, the Simulation Object Model (SOM) which defines its data requirements from the simulation network and the data it can provide to the simulation. The union of these SOM's is the Federation Object Model (FOM). The federates communicate using a distributed network system called the Run Time Infrastructure (RTI).

The High Level Architecture is the formal means for making the federation work together. It consists of three parts:

1. The ten HLA Rules which specify how the Federates communicate in the Federation,

2. The Interface Specification (IF) with specific functions that the RTI must provide and the Federates must use to communicate, and

3. The Object Model Template (OMT) which specifies the common data formats used by individual Simulation and the Federation object models. [1]

DMSO certifies individual federates and their SOM's as being HLA compliant. To be certified, they must follow the ten HLA rules, their communications must adhere to the Interface Specification, and their Object Model must conform to the OMT. Because a standard FOM and SOM's are not required, interoperability between compliant federates is not automatic. [1]

There are three paths to HLA implementation. The first is the " Native," or full code compliance. The second is the "Gateway," or conversion of the existing output stream, usually DIS, to an HLA compliant stream, by a Gateway Processor. The third is "Middleware," or "Internal Gateway," where the internal data stream, usually DIS, is converted to HLA before being placed on a network. Figure 3 HLA Implementation Strategies demonstrates these concepts. [4]

The "HLA Interface Ambassadors" in each method in the figure are the DMSO implementation of the communications requirements of the Interface Specification and HLA Rules. The "FED Ambassador is federate code, written to the RTI standard, that accepts transmissions from the RTI. The RTI Ambassador is RTI code that the federate calls to send messages to the RTI.

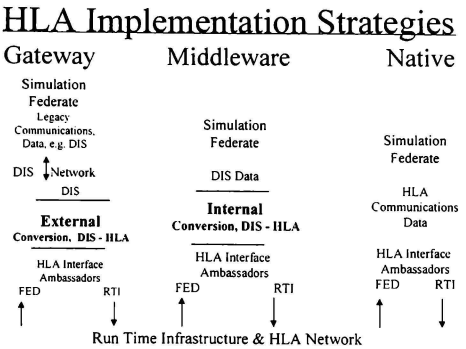


Figure 3 HLA Implementation Strategies [4]

Native

The native approach is the fastest, because only one processor is involved. It is generally the most expensive for legacy application conversion, because it requires extensive revision of the software to comply with the RTI Interface Specification and HLA communication rules. It is only appropriate for new simulations.

Gateway

The gateway approach is the least expensive, as it requires only another processor in the communication stream and no new programming. This is also the slowest, as the additional processor introduces delays in sending data to and receiving data from the network. In cases where no RTI exists for the simulation platform, gateway mode is required.

Middleware

The middleware approach is almost as fast as native, because it does not require another computer in the communication link. It is more expensive than gateway as custom programming is required to integrate the middleware code and the simulation code.

How it works, really...

The use of HLA technology must mature through experience before the benefits of the DOD and DMS interoperability vision can be realized.

The Naval Air Systems Command (NAVAIR) describes its plans using the Interoperability Maturation Model in Figure 4.

As the Interoperability of Naval Aviation Simulations matures through the five levels, the HLA implementation strategy matures from gateway to middleware to native, and the data use matures from object models for existing DIS systems to scenario generation on demand. This maturity allows the mission rehearsal and distributed training applications to become possible.

The Simulation and Federation Object Models (SOM, FOM) are the collection of all data required by and available to the federates in a federation. In practice, this can result in many different, and expensive, SOMs and FOMs.

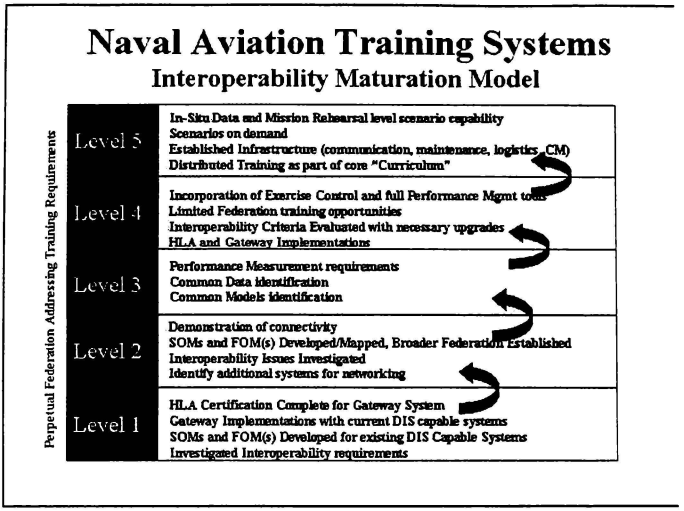


Figure 4 NAVAIR Interoperability Maturation Model [4]

This led to efforts to standardize Object Models in order to reduce repetitive development expenses, and to take advantage of existing simulations. This has resulted in the development of standard, or reference FOM's.

One of the most prominent such reference FOMs is the Real Time Platform Reference Federation Object Model (RPR-FOM). This is based upon translating DIS standard PDUs to HLA. The authors present as one justification for its development the "*a priori* interoperability" of an HLA federation through its use. [5]

The use of common validated, verified, and accredited models, data, and simulations allows for the use of computer generated forces to play the opposing force (OPFOR) roles, as well as to augment the force being trained. These models participate in the exercise in a technically and doctrinally correct fashion. Behavior modeling allows the simulations to appear to be judgmental in their responses to tactical situations, and not merely present a predictable, automatic reaction to the situation that would result in negative training to the live forces.

Examples

Table 3 Example SMOC - HLA Federates summarizes NAWCTSD experience with SMOC in building gateway and middleware HLA federates. No native federates have been built by NAWCTSD. They would not be built unless a complete software reprogramming effort was required, which would be a contracted effort. The federate name, a short description, implementation type, point of contact, and DMSO certification status, if any are given.

The DMSO maintains a reference library of certified HLA federates, points of contact, and the Simulation Object Model used for the certification. This is available through the DMSO HLA web site, http://hla.dmso./mil. [1]

Since the middle 1990's, the Naval Air Warfare Center Training Systems Division (NAWCTSD) has developed a combination Gateway/ Middleware HLA - DIS package named the Simulation Middleware Object Classes (SMOC). The work has been sponsored by a series of research grants and applications to various projects. Table 3 Example SMOC - HLA Federates

lists the NAWCTSD efforts using SMOC and, when applicable, their DMSO certification status. [1,2,3]

Table 3 Example SMOC - HLA Federates

FEDERATE NAME	DESCRIPTION (FOM), SPONSOR	TYPE	NAWCTSD POC	DMSO CERT.
F-14	RPR ONR	Middleware	Kevin Geib Dan Paterson	JAN 98 DEC 98
Multi Mission Tactical Trainer (MMTT)	MMTT - CIC NAVSEA PMS-430	Middleware	Kevin Geib	NOV 99
CSMET: Kiowa Warrior OH-58	CSMET (RPR) Army Sponsor	Gateway	James Miles, Erik Hougland, Dan Paterson	MAY 2000
Modular Semi-Automated Forces (ModSAF) 4.0	RPR Army Sponsor	Middleware	Erik Hougland	

The SMOC is based upon a set of "Channels" that accept DIS, HLA, and application input and output. In Gateway mode, DIS channel - HLA channel conversions are performed. In middleware mode, host application channel - HLA channel conversions are performed. The DIS channel is available if gateway services are also required.

The SMOC is available commercially from Distributed Simulation Technology, Inc. (DiSTI), Orlando, FL. (http://www.simulation.com). DISTI is responsible for commercial distribution and maintenance of the package.

The key to middleware integration of SMOC is the identification of four interface function points in the host code:

DIS input
DIS Output, and
The Host code timing loop.
Simulation and Network Initialization

Middleware Example

In the ModSAF 4.0 HLA compliance project, a proof of the middleware concept was desired. The million, or so, lines of ModSAF code were analyzed to find the interface function points described above. These points are listed in Table 4 SMOC - ModSAF Interfaces and illustrated in Figure 5 ModSAF - SMOC Architecture from the SMOC - ModSAF project report and the Simulation Interoperability workshop paper on the project. [3]

Table 4 SMOC - ModSAF Interfaces

SMOC Interface Function	Point in Figure
DIS Input	Smoc_Read
DIS Output	Smoc_Write
Timing Loop	Smoc_Called_Each_Frame
Initialization	ModSAF Initialization

Once the points were identified, C++ functions were written to link the ModSAF and SMOC functions. The proof of concept project was considered successful, but additional work is needed to extend it to the current ModSAF, version 5.0, and to successor computer generated force products. [3]

Gateway Example

The third phase of the Multi Mission Tactical Trainer (MMTT) and Battle Force Tactical Training (BFTT) systems is illustrated in Figure 6 MMTT / BFTT Gateway and Middleware Example. SMOC is used in both gateway and middleware modes. The "G" legend indicating a gateway is SMOC in gateway mode. The BFTT and MMTT applications in the circles use SMOC operating as middleware.[2]

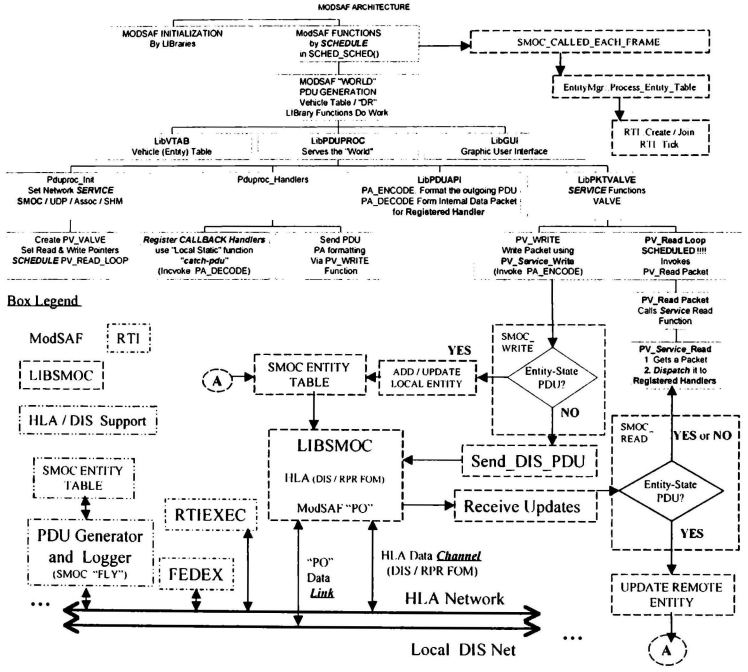


Figure 5 ModSAF - SMOC Architecture [3]

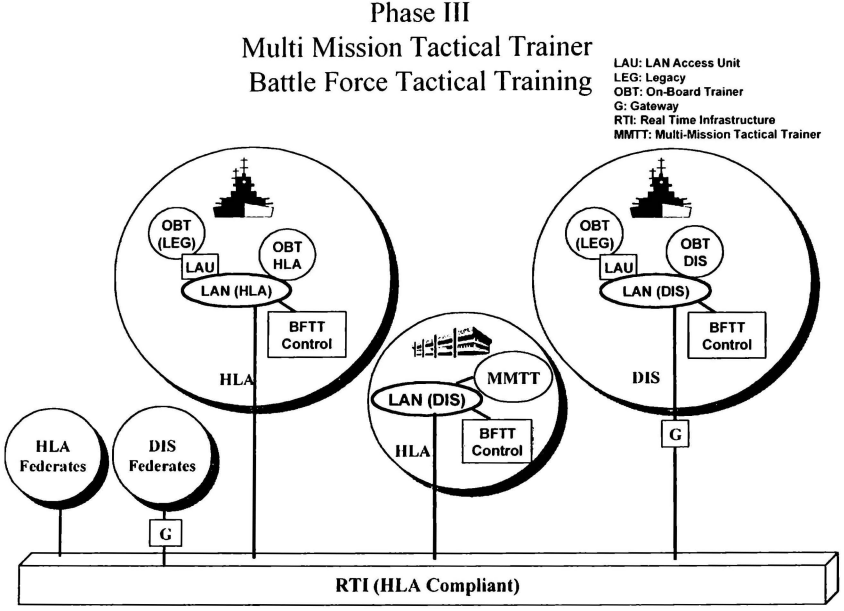


Figure 6 MMTT / BFTT Gateway and Middleware Example [2]

Lessons Learned

The SMOC is a useful tool for both Gateway and Middleware HLA compliance.

SMOC is capable of being used in both gateway and middleware in major HLA-DIS integration projects, at various phases in the lifetime of the project.

SMOC gateway and middleware implementations are significantly cheaper than Native mode. For example, the CSMET Gateway effort has used less than 400 Work Hours, or about one quarter work year. Native conversion was not formally estimated, but could easily exceed a work year.

The SMOC is a viable commercial alternative for HLA implementations as it is supported commercially via the CRDA with DISTI. The increased exposure of the SMOC will allow its improvement in response to customer experience.

The identification of the integration points can be a tedious, circuitous, task in a large program system, such as ModSAF. Once identified, the work progressed in a straightforward fashion.

There are other HLA Gateway packages available. A notable product has been the Army Simulation, Training and Instrumentation Command (STRICOM) Gateway. STRICOM recently terminated its funding of the project. [6] The product is now available as the MAK Gateway. MAK is also the vendor of VR-LINK, a network interface package that can have one output channel open at a time, either DIS or HLA. Other gateway products are under development through small business innovation research programs (SBIR).

References

1. Defense Modeling and Simulation Office High Level Architecture Web Site, http://hla.dmso.mil.

2. Geib, Kevin K., Multi Mission Tactical Trainer and Battle Force Tactical Training Project, Personal Communications, January-June, 2000.

3. Hougland, Erik S. and Paterson, D.J., Demonstration of High Level Architecture (HLA) Middleware Technology Integration with a Legacy Application Architecture, Paper 99F-SIW-057, Fall Simulation Interoperability Workshop, Orlando, FL, September, 1999.

4. Paterson, D. J., and Fisher, Rory H. "Naval Aviation Training Systems Maturation Model, Paper S361, Summer Computer Simulation Conference, (to be presented) July, 2000.

5. Reilly, Sean (ed.), Guidance, Rationale, and Interoperability Modalities for the Real-time Platform Level Reference Federation Object Model (RPR-FOM), Version 0.4/0.5, August, 1998,

6. MAK Industries, www.mak.com

Authors

ERIK S. HOUGLAND is an electronics engineer with the Naval Air Warfare Center Training Systems Division, Battle Force Simulation Branch. He is a member of the NAWCTSD Interoperability Laboratory, with research papers on HLA implementation *via* middleware and the incorporation of advanced HLA services in middleware packages. He has authored studies on strategy for and costs of implementing HLA in Naval Aviation training systems.

In '1973 and 1977 he received the MS and Ph.D. from Virginia Polytechnic Institute and State University with research on applications of computer science and operations research to environmental systems. He received the B.S. in Mathematics from Rensselaer Polytechnic Institute, Troy, NY, in 1966.

In 1966 he received his commission in the United States Navy. He served on active duty in Diesel Submarines and in Radio Link Battlefield Electronic Instrumentation in The Republic of Viet Nam. He completed his Naval Reserve career as a scientific and technical intelligence officer

DANIEL J. PATERSON is an electronics engineer with the Naval Air Warfare Center, Training Systems Division, Simulation & Models Division, Modeling & Simulation Development Branch in Orlando Florida. He is currently the technical leader of the NAWCTSD Interoperability Laboratory, with research papers on HLA Gateways and Data Distribution Management.

In 1992 he received his MSEE degree from the University of Central Florida. In 1976 he received his BA in Mathematics from Jacksonville University, Jacksonville Florida.

In 1977 he received his Naval Flight Officer Wings from the US Navy. He spent 8 years flying in Naval Patrol Aircraft, before transferring to the US Naval Reserves where he spent another 12 years flying as a Reserve Officer for the US Navy.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37276

AIAA-2000-4487

Joint Strike Fighter Virtual Air-to-Air Combat Analysis For the Joint Operational Requirements Document

S. T. Bishop, T. E. Menke, D. R. Mittlesteadt
United States Air Force/Aeronautical Systems Center
Wright-Patterson Air Force Base, Ohio 45433

ABSTRACT

The Joint Strike Fighter (JSF) program conducted two virtual air-to-air combat effectiveness Aircrew System Advisory Panels (ASAPs) at WPAFB in support of the Joint Operational Requirements Document (JORD). In December 1998 the program sponsored a collateral-SECRET level simulation to evaluate possible avionics configurations. Trade studies were conducted on proposed radar, infrared, and other weapons system attributes to determine their impact on JSF combat effectiveness in the Defensive Counter Air mission. Pilots from the JSF Operational Advisory Group (OAG), tasked to write the JORD, participated in the event.

In September of 1999, Simulation and Analysis Facility (SIMAF) conducted an additional virtual event to further explore the concepts and tactics of the Close Escort and the Self Escort mission scenarios. The focus of the event was to allow the foreign partners of the JSF program to evaluate the performance of the JSF with their legacy aircraft, the F-16. Concept trade studies included various levels of threat and threat capabilities, and for the first time the addition of a surface-to-air threat.

The simulations featured medium to high fidelity modeling of radar, infrared, data fusion, and weapon system effects. Pilots flew both JSF and threat aircraft to explore tactics and CONOPS. Analysts collected data to evaluate weapons system performance and gather tactical concepts to guide in the future employment of the weapon system.

This multimedia presentation will feature the important UNCLASSIFIED findings of the analytic results. The results have particularly significant implications to the modeling and performance of sensor fusion systems. Finally, the presentation will comment on the role of air-to-air operator-in-the-loop modeling in simulation-based acquisition.

INTRODUCTION

In 1994, the Secretary of Defense, Secretary of the Navy, and Secretary of the Air Force to begin an acquisition program with a focus on simulation chartered the Joint Strike Fighter program. Specifically, the JSF program was to facilitate a fully developed & validated set of operational requirements. To accomplish this an integrated team of users and developers was formed to address these requirements. The charter further stated the need to conduct tradeoff analyses of critical user defined performance parameters using unprecedented levels of joint analyses and simulation and evolve the requirements over time.

To respond to this charter the JSF Program Office began to evaluate what modeling and simulation capability was available to do the job. Under the guidance of the Simulation Based Acquisition principles, they formulated a complex requirements generation and validation plan. Program decision makers knew the importance of including the warfighter in the modeling and simulation process. For the air-to-air role for the JSF, the program came to MIL-AASPEM, Man-in-the-Loop Air-to-Air System Performance Evaluation Model. This allowed the Operational Advisory Group to become immersed in a synthetic environment to more accurately understand the requirements and test the parameters of those requirements in a simulated aircraft cockpit.

The Joint Strike Fighter (JSF) Program is unique in that it is weaving together multi-service requirements (Air Force, Navy, Marines, and foreign participants) while maintaining a single, joint system program office. Each ASAP event supported the requirements definition effort by providing objective analytical results to the JSF community, as well as by providing Operational Advisory Group and foreign participant pilots with virtual "hands-on" experience that can be used to further define program goals and requirements.

Figure 1 illustrates the WPAFB JSF Aircrew Systems Advisory Panel Team. The team is comprised of a large number of contractors and government engineers. Any given subtask is comprised of a government lead supported by both government and contractor engineers of various organizations and technical disciplines. These efforts are coordinated by the simulation and analysis leads within AFRL/VA and ASC/ENM. Overall project coordination is provided by the local JSF Program Office, ASC/FBJ.

This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

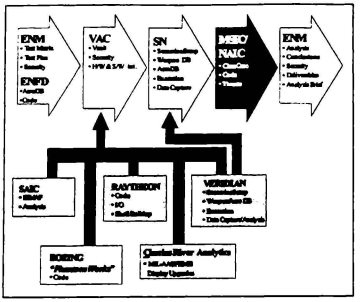


Figure 1 WPAFB Air-to-Air Team

This federated team of technically diverse engineers scientists, and analysts brings WPAFB to the forefront of modeling, simulation, and analysis. The people attached to this federated team are the key to the success of the Simulation and Analysis Facility (SIMAF) at WPAFB. The strategic alliances formed among the team provide the most accurate and current threat system data available, highly technical analytical capability, and complex hardware and software design and development. Moreover, the capabilities will assist the JSF program to realize the Simulation Based Acquisition goal set forth for the in their charter.

SIMULATION AND ANALYSIS FACILITY

During the early development of SIMAF, the existing simulation facilities provided significant input into SIMAF definition, including required hardware, software, physical layout, and network connections. SIMAF would need presentation facilities for 100 people, floor space of over 40,000 square feet, segregated areas for classified work, connectivity, analyst workstations, and piloted simulators.

Two-sided virtual wargaming-simulation and analysis can be done at the campaign level. SIMAF accommodates the Thunder model, in both real-time and compressed modes. Two "Team Areas" are provided; each "Team Area" accommodates 8 analysts, with the ability to isolate the two areas from each other. Team Areas have flexible computer processor and mass storage access, visual line of sight and direct audio contact with the video wall and simulation director areas. Each Team Area has a commander's station that includes a computer generated map table. The map table is used for monitoring analyst workstation displays, stealth viewing, moving map displays, and mission planning displays. Each map table has a capability for 2D stealth viewing, playback and recording.

Connectivity to numerous Wright-Patterson AFB assets has been demonstrated including the ASC Major Shared Resource Center (MSRC), AFRL/CFH, AFRL/AAS, and ASC/ENA. Connections to external assets (outside of Wright-Patterson AFB) have also been demonstrated. These include facilities at the Air Armament Center, Eglin AFB such as PRIMES and GWEF, and the "math lab". The Bennefield Anechoic Facility at Air Force Flight Test Center, Edwards AFB and the Air Combat Environment Test and Evaluation Facility (ACETEF) at the US Navy's Patuxent River Naval Air Station are planned locations for distributed simulations within the first year of operation. Other general wide-band connections exist and were used in 1999 to participate in EFX99 that included live, airborne test assets.

ASC, including all of Wright-Patterson Air Force Base, had no general purpose "special access" and "sensitive compartmented information" processing facility. Additionally, there had been limited capability for multiple, distributed real-time simulation coordination and control. SIMAF has resolved this shortfall in a big way. Up to 6 simultaneous yet isolated highly classified simulation events can be conducted in SIMAF. Any combination of connections between these events can also be realized as required by the various programs.

As the requirements for the facility were defined, the engineering and analysis team realized a "real-world" project would help rapidly identify and size the hardware and software resources that SIMAF would need. Through a series of meetings, the Joint Strike Fighter (JSF) program was identified as the ideal first program to be simulated in SIMAF and the first start-to-finish test of the engineering and analysis team's concept of how the air combat MS&A federation should function. Both air-to-air and air-to-ground simulation exercises were conducted for JSF.

PILOTED AIR-TO-AIR SIMULATOR

For air-to-air combat simulation the facility contains eight pilot stations with expandability up to twelve, two AWACS stations, and one simulation director console. Each pilot station is powered by a Silicon Graphics O2; while the simulation control station is on a 4-processor desk-side ONYX (see Figure 2). The two sides (red and blue) of the scenario are separated to limit cheating

and gaming the scenarios. Each side has intra-flight digital voice communication including the AWACS controller.

The main simulation engine is MIL-AASPEM (Man in the Loop - Air to Air System Performance Evaluation Model) II, or MIL2. MIL2 is a real time, interactive, many-versus-many model to calculate the engagement of multiple platforms. It is used to conduct fighter weapons system effectiveness analyses, combat tactics development and requirements analyses. MIL2 is used primarily for air-to-air analyses, but has some surface-to-air and air-to-ground capability. Aircraft in the simulation can be flown either with the piloted stations or can be fully computer driven. The simulation can also run in a batch mode to evaluate numerous

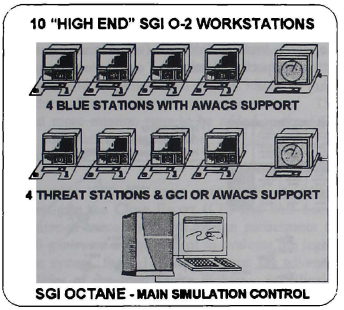


Figure 2 SIMAF Piloted Air Combat Layout

scenarios and decision matrices. Another part of the model (AMFOM) can be run in a 1v1-footprint generation mode to generate missile launch envelopes, sensor envelopes, counter measure envelopes, and many other factors of interest. Originally a Boeing Product developed with both government and Boeing funding, MIL2 was entered into SURVIAC last summer and is available to all government organizations.

The JSF program became interested in the capabilities that the MIL-AASPEM model provided to investigate the performance of the simulated JSF aircraft against a manned threat weapon system. Focusing in on the many against many functionality and the level of fidelity available for the aircraft sensor functions, the program included the ASAP events in the evaluation of the weapon system requirements. JSF has assured that a "leave behind" capability for future weapon systems will be present through their funding support.

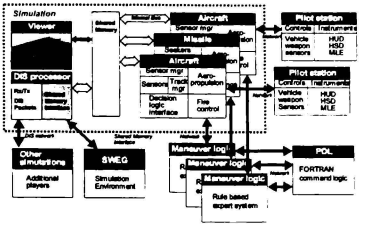


Figure 3 MIL2 Architecture

The main simulation consists of an object oriented pseudo-real-time programmable cyclic executive and works through shared memory and networking. The main simulation process runs on a single R-10000 CPU. The main simulation process includes the aircraft, missile, ground site objects as well as the simulation executive (see Figure 3). Pilot stations, low-level Piloted Decision Logic (PDL), and the higher-level (Realist) decision logic processes are run on other additional CPU's. This allows additional players to be modeled in separate player processes on the main simulation node while the PDL and Realist processes to control computer-generated players are executed on additional CPUs. Piloted players require an additional CPU (for the pilot vehicle interface). In SIMAF, each pilot stations was run on a single SGI O-2. The pilot station consists of a display process and a Hands-On-Stick-And-Throttle or HOTAS process.

The simulation viewer interfaces with the simulation through shared memory in a Director mode to control all the simulation process and the Gods-Eye Viewer providing an overview of the Battle Space (see Figure 4). The MIL2 Viewer, the Virtual Battlefield Management System (VBMS), is also a stand-alone tool developed by Air Force Research Laboratories. VBMS is a multi-window stealth viewer providing a scalable moving map mode, 360-degree azimuth and 180-degree elevation 3-D object viewer, and a user-friendly control panel. Additional viewer options include VCR playback and record capabilities, snapshot, player filter options, terrain detail options, player entity data, path display, orientation displays, and numerous additional simulation controls. All tools also can be brought up in an Object Browser mode to browse the internal object hierarchy.

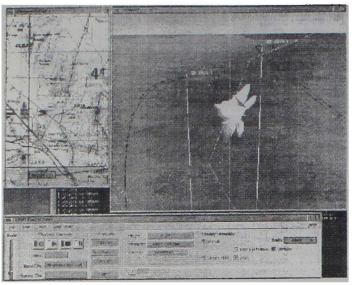


Figure 4 MIL2 Controller Display

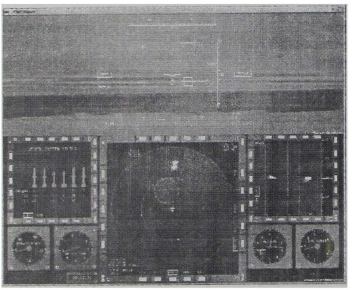


Figure 5 MIL2 Pilot Station Display

The Realist (Expert System) Process has a browser to interrogate the rule bases. The pilot station includes a hands-on stick and throttle (HOTAS) for pilot input and an OpenGL based pilot cockpit and out-the-window display complete with HUD (see Figure 5).

The Realist processes perform the deterministic rule-based own-ship decision logic for each player in the simulation. The Pilot Decision Logic process performs the global decision logic for group tactics.

MIL-AASPEM's test director/simulation node is networked to the other distributed processes across a network through programmable data packets. In addition it can be connected to a Distributed Interactive Simulation (DIS) network for interaction with other DIS simulations or it can interact with the Joint Interim Mission Model (JIMM) through a shared memory interface. In the future MIL-AASPEM will be compatible with the High Level Architecture (HLA) widely being accepted as the standard for distributed simulation activities.

VIRTUAL SIMULATION EVENTS

The WPAFB JSF virtual simulation team executed its first air-to-air event in June of 1998. The event supported the development of the third Joint Interim Requirements Document or JIRD-III, the last interim document before the release of the draft JORD.

The air-to-air event investigated the use of the Joint Strike Fighter (JSF) in a Defensive Counter Air (DCA) scenario with four JSFs opposing four threat escorts and four threat strike aircraft. Each side used an airborne early warning aircraft (see Figure 6) to support their respective missions.

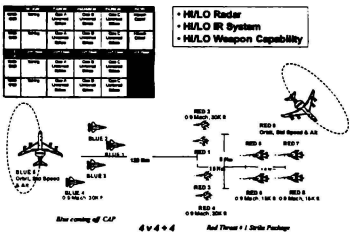


Figure 6 ASAP Test Matrix and Scenario Setup

The test matrix investigated a number of radar, infrared, and weapon attributes for the JSF DCA mission. The test matrix is a Design Of Experiment (DOE) matrix of resolution four. A resolution four DOE allows for unconfounded, independent primary attributes. The attributes selected for investigation during this ASAP are determined based on the Operational Advisory Group's requirement for data to support the development of the JORD. The OAG provided the analysis community with a large number of questions that required analytical underpinnings to support the development of the JORD. Each attribute selected for incorporation into the test matrix is correlated against the applicable OAG questions to ensure traceability within the matrix. (See Figure 7)

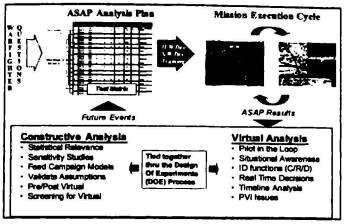


Figure 7 ASAP Analytic Process

Later in December 1998, JSF continued to focus their efforts on the defensive counter air mission scenario. In order to build on the success of the June ASAP, the mission scenarios were very similar. Again design of experiments techniques were employed to ensure that the most could be gained from the test results to guide any future efforts in the requirements process. This allowed more mission runs to be added to the DCA study and further support the findings from the previous ASAP efforts.

In September 1999, JSF again went to MIL-AASPEM to investigate the Close Escort (CE) and Self Escort (SE) mission scenarios. The foreign participants for the JSF generated a requirement to evaluate the legacy weapon system, the F-16 MLU, against the advanced capabilities of the Joint Strike Fighter. Specifically, the functional objectives are to investigate the following:

- Various Threat Systems Sensitivities (Missiles, Radar, etc.)
- Surface-to-Air Threat Sites

BLOCK	Threat GCI	Mission	Blue Aircraft
A	A	CE	JSF
B	A	CE	F-16
C	A	SE	JSF
D	A	SE	F-16
E	P	CE	JSF
F	P	CE	F-16
G	P	SE	JSF
H	P	SE	F-16

*A – Absent CE – Close Escort
P – Present SE – Self Escort

Figure 8 ASAP Analytic Test Matrix

Figure 8 shows the executed test matrix to support the interests of the OAG for the event. As indicated, the same missions will be flown using both the F-16 and the JSF to provide a direct comparison in the mission scenario. The results of this study will provide insight to foreign participants and to the JSF OAG regarding key JSF operational and system requirements and tactics.

In the CE mission, two JSF aircraft provided close escort for four JSF strike aircraft (See Figure 9). All six blue aircraft were manned. The target for the strike aircraft was a highly valued bridge. The Blue players to penetrate a threat Surface-to-Air Missile (SAM) belt in order to reach the assigned target area. Two Red threat aircraft initially performed Combat Air Patrol (CAP). These players came off CAP to oppose the JSF as they begin to penetrate Red airspace. Additionally, two Red players were scrambled from a nearby airbase to oppose the JSF on their egress from red airspace. The exact time at which these threat aircraft are scrambled varied depending on the test block.

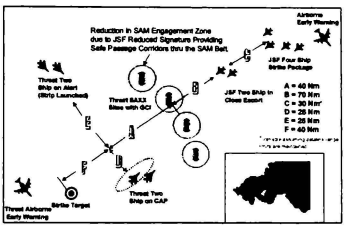


Figure 9 ASAP Close Escort Scenario

The SE mission was similar to the CE mission, except that four JSF aircraft performed a self escort into red airspace in order to attack the bridge strike target (See Figure 10). Trained aggressor pilots flew the simulated threat aircraft cockpits. Current operational pilot representatives from the OAG and foreign participants flew the JSF simulated aircraft. The mission configuration was different for the SE mission. Instead of the 6 vs. 4 in the CE mission with 4 of the JSF loaded with only air-to-air weapons, the SE mission focused on a 4 vs. 4 scenario and each JSF aircraft had both air-to-air missiles and air-to-ground bombs.

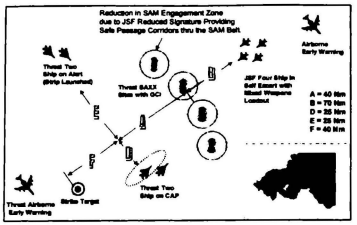


Figure 10 ASAP Self Escort Scenario

Two days were spent on training during each week of each event. Pilots were briefed on the scenario, test matrix, analysis objectives, JSF modeled systems, threat systems, weapons, and pilot vehicle interface limitations. Training results were recorded to provide insight into the training effectiveness and to measure the learning curves of all of the pilots. This process was intended to level the skill set of the participating pilots to decrease the variance incurred from the differing levels of experience among the pilots.

The execution of the test allowed each side to plan their missions, execute those missions in the simulation, and debrief those missions using the test director (see Figure 7). Each simulation run was replayed immediately after execution to allow pilots to review and critique their actions prior to completing a detailed pilot questionnaire. The pilot questionnaire was designed to capture subjective data not readily available within the objective simulation data. "Hot washes" are completed at the end of each day and an overall "hot wash" at the end of the week to allow the analysts and warfighters an opportunity to discuss the simulation results and provide final feedback on both the test results and any simulation pilot vehicle interface (PVI) issues.

ASAP RESULTS AND IMPACTS

The results from simulation events, conducted in support of the JSF JORD, have altered the quantity and quality of information available to the warfighter, analyst, and decision maker. The use of virtual simulation in the Joint Strike Fighter program has allowed the immersion of the warfighter in the proposed technology during the development of the requirements. This immersion significantly alters the warfighter's perspective in developing tactics to exploit the new system capabilities. Further, the warfighter gains insight into the requirements process not available by other more conventional approaches. The use of virtual simulation is essential in determining more esoteric but important attributes of the system such as situational awareness during a combat mission. Additionally, timeline analysis provides insight into the criticality of the arrival of off-board information to successfully survive or complete a mission. Virtual simulation also categorizes pilot perceptions and decisions regarding the classification, recognition, and discrimination of threats and/or targets during various missions and mission phases. Further, the use of virtual simulation is necessary for the employment of any subsystem requiring a real time decision from an informed pilot. These type of decisions are common for a pilot when employing nearly every sensor on a modern fighter.

The objectives from each of the ASAP events were met and exceeded. Going into the release of the JORD, the JSF program had very good insight into the Defensive Counter Air mission scenario, as well as the Close Escort and Self Escort scenarios. The June 1998 trade between the two levels of radar and infrared sensor capabilities allowed the program to balance the capabilities with the whole weapon system in mind. Also, it showed the impact that the command and control architecture had on the success of the mission. As an integrated system of systems, the JSF not only benefits from the system, but also is an integrated provider of information about the threat area.

The investigation in December 1998 continued along the same vein of the DCA mission scenario. The JSF program gained much more insight into the requirements for successful completion of the mission scenario. Continuing to do analysis trades in the avionics capabilities, the JSF program benefited from the information from the June event and focused its efforts on the generated results.

These two ASAP events allowed the OAG decision makers to refine the DCA requirements for the JORD and provide a more defined documents to feed the remainder of the acquisition program cycle. As an added benefit, the pilots were able to use the planned advancements in the JSF to define the tactical concepts that will be used in the fielded JSF system.

For the September 1999 ASAP event, the program wanted to focus in on the CE and SE mission scenarios. The objectives as defined allowed the WPAFB analysis team to generated complex and detailed probability of lethality and survivability data. Moreover, the timeline analysis gave the OAG decision makers much more insight into the mission critical timing of a combat mission. For the foreign members of the OAG, the direct comparison of their legacy F-16 and the

proposed JSF allowed the pilots to provide critical data back to their respective home countries for them to evaluate their requirements for the JSF. Also, the information gave the foreign decision makers enough information that they will continue to support the acquisition of the JSF and consequently include the advanced fighter in their own combat air force.

With each ASAP event, sound statistical principals are employed to gain as much information from the data as possible. For each test matrix, complex design of experiments techniques were used to make each test block as complete and valuable as possible. The only limitation is the amount of data runs that are possible from each event. Due to limited time and pilot availability, only a few test runs can be completed for any of the test blocks (typically, fewer than 10). Further, analysts can only infer simple trend data from any of the test blocks. No statistical confidence can be gained from the data with so few test block runs. Therefore, virtual man-in-the-loop simulation cannot stand alone in the insights required for the JSF program. The SIMAF analysts rely on the constructive capabilities to drive the sample size up for the test matrix. A collaborative effort is ongoing to incorporate the virtual data into the constructive models and accomplish sufficient test runs to determine the statistical significance and draw definitive conclusions. As a rule, analysts caution decision makers on drawing any decisive conclusions based solely on virtual simulation data.

The warfighter is not the only beneficiary from virtual simulation. The decision-maker can observe the application of the system in a mission environment as employed by the warfighter. This tremendous benefit provides the decision-maker with the credible application of the technology or system removed from editing by intermediate players. Further, the endorsement by the warfighting community provides additional credibility to the virtual simulation results. Other beneficiaries include the weapon system contractors (WSCs) who both witness the events and receive analysis data and pilot vehicle interface (PVI) comments from the Navy, Air Force, and Marine Corps pilots participating in these events. The results are used to further refine the contractors' future virtual simulation events, ensuring a more robust, credible, and worthwhile simulation event. Additionally, many military organizations request data from these events to support ongoing research, current programs, analytic development, or as support for simulation based acquisition (SBA) initiatives. Other modeling simulation organization are looking for a capability to include high fidelity manned cockpits for inclusion into their simulation and testing concepts.

THE ROAD AHEAD

Future efforts are focused on addressing modeling deficiencies identified in previous events. The incorporation of new OpenGl displays and test directors provides a new user-friendly look and feel to the MIL2 model. The newest version of MIL2 also incorporates many new subsystem functions with the accompanying displays such as the ability to expand any location on the radar, improved Missile Launch Envelope (MLE) algorithms, improved heads up displays, improved heads down display features, and dynamically recorded playback mode. Further planned improvements will incorporate threat specific displays, a Higher Language Architecture interface, a Distributed Integrated Air Defense System (DIADS) interface, and the addition of new Pilot Decision Logic algorithms incorporating new Artificial Intelligence (AI) techniques.

These advancements will allow more flexibility in the use of the air-to-air mission scenarios for any other weapon system to evaluate technological advancements. With the MILII simulators, any program can integrate their model and simulators in an environment that will provide either a command and control node or threat aircraft systems. Those programs can then interface with the added variance that a real operator brings to the combat picture. This has much more advantages over the use of computer-generated simulated players. Additionally, more advancement in the models used by MILII will allow for the inclusion of higher fidelity players that will more closely reflect the actual weapon systems in the combat environment.

To build on the connectivity capabilities that SIMAF provides, the MILII simulators are involved in several efforts to include manned threat cockpits in other modeling, simulation, and analysis efforts no matter where those efforts are located. This effort will expand the capability that other MS&A facilities possess to include a much more realistic threat to encounter for their weapon systems tests.

Another effort that the WPAFB Air-to-Air Team is concentrating on is the transition of the MIL-AASPEM cockpit stations into the personal computer architecture. The effort is to port the simulation software in the LINUX operating system and use lower cost computer hardware to drive the simulation. This will open up the capabilities that MILII brings to a weapon system program to a much broader customer base. With today's defense budget declining, acquisition programs are forced to find less expensive ways to test and evaluate their concepts or designs in

an environment that will exercise the full potential of the system.

The Joint Strike Fighter program has included the MIL-AASPEM model in their core suite of models and analysis tools. This core suite is slated to fully support the next phase of the acquisition cycle; Engineering, Manufacturing, and Development. The models will continue to integrate and develop to support a consistent environment to evaluate the JSF weapon system throughout the later phases of the program.

With all the plans in place, the Man-in-the-Loop experience will continue to grow in importance to any new acquisition program that will start in the future. Thus, allowing the total acquisition cost to be lower and allow all of the benefits that a simulated mission environment brings to the warfighter and the decision makers.

SUMMARY

The Joint Strike Fighter program has responded to the Department of Defense charter to include complex and integrated modeling and simulation in the acquisition cycle of the program. The three ASAP events proved to add significant value to the JSF Operational Advisory Group and their understanding of the future role of the aircraft. The studies performed illustrated that WPAFB and SIMAF have fully supported the requirements definition process and was integral to the development of the JORD for JSF.

At WPAFB several diverse but related offices have joined to form a new modeling, simulation, and analysis federation. This federation is capitalizing on their human and computer resources to reinforce the application of solid systems engineering and analytical processes upon early acquisition efforts for military modernization.

The use of virtual simulation in support of the Joint Strike Fighter's JORD provides a prototype for both WPAFB and the Air Force in applying SBA concepts to DoD requirements analysis. The use of virtual simulation both within the Air Force and Industry will continue to increase and expand in scope as the application of virtual simulation matures.

There will remain focused efforts to improve the MS&A capabilities of SIMAF and other organizations and to provide highly flexible simulators to support acquisition testing. This will continue to move the MIL-AASPEM model into the forefront of SBA activities for JSF as well as, other new and emerging weapon systems.

The WPAFB air combat MS&A federation is uniquely qualified to provide the front end of the SBA approach and is pushing the simulation environment to complete the SBA circle with the flight test and training communities.

REFERENCES

1. Rodgers, S.M., et.al, Gansler, J. S. Under Secretary of Defense Acquisition and Technology, "Life-Cycle Modeling and Simulation A Key Element In Acquisition Reform," National Defense Industrial Association, Simulation-Based Acquisition Workshop, Orlando, Florida, March 17, 1998.

2. Moorman, T. S., Jr., General, USAF, Keynote speech to the Interservice/Industry Training System and Education Conference, Albuquerque, N.M., "Today's Technology, Tomorrow's Training," Nov. 14, 1995.

3. Sanders, P., Ph.D., Director, Defense Test, System Engineering and Evaluation, "Simulation Based Acquisition: An Effective, Affordable Mechanism for Fielding Complex Technologies," March 21, 1997.

4. Aeronautical Systems Center Simulation and Analysis Facility (SIMAF) Requirements Document, 1997

5. Menke, T. E., Briefing to the Defense Planning and Analysis Society, "The Use of Virtual Simulation in Support of the Joint Strike Fighter's Joint Operational Requirements Document (JORD): A practical Simulation Based Acquisition (SBA) example," April 14, 1999.

6. Pew, R. W. and Mavor, A. S., Eds, Modeling Human and Organizational Behavior: Applications to Military Simulations, Washington D.C., National Academy Press, 1998.

7. Joint Strike Fighter Aircrew System Advisory Panel Air-to-Air 1 Analysis Plan, Aeronautical Systems Center, Virtual Simulation Group, 1998.

8. Joint Strike Fighter Aircrew System Advisory Panel Air-to-Air 2 Analysis Plan, Aeronautical Systems Center, Virtual Simulation Group, 1998.

9. Joint Strike Fighter Aircrew System Advisory Panel Air-to-Air 3 Analysis Plan, Aeronautical Systems Center, Virtual Simulation Group, 1999.

10. Rodgers, S.M., Hellmann, G.K., Richart, D.K., Mittlesteadt, D.R., Zeh, J.H., and Menke, T.E., "The "New" Air Combat Modeling, Simulation, and Analysis Federation at Wright-Patterson AFB, OH," May 1999.

11. Man-in-the-Loop Air-to-Air System Performance Evaluation Model (MIL-AASPEM) User Guide, Rev A, September 30, 1998.

A00-37277

AIAA Modeling and Simulation Technologies Conference and Exhibit 14-17 August 2000 Denver, CO

AIAA-2000-4488

Using Virtual Simulation for Tactics Development for Constructive Simulation

Stephen E. Thomas
AFRL/IFSD, WPAFB, OH

Cindy Martin
Veridian, Dayton, OH

ABSTRACT

The development of tactics to exploit the advantages of new systems is one of the most important uses of virtual simulation. Due to resource constraints, virtual simulation is rarely used to perform exhaustive testing, and constructive simulation must be used to get statistically significant results. Translation and coding of tactics developed in virtual simulations into constructive simulations is a very difficult problem for analysts to perform. There are things that can be done during virtual testing which can make this process go more smoothly. Having a good questionnaire can help the analyst translate what the pilot did during the simulation in the constructive analysis tool. Most constructive tools have basic offensive tactics, defensive tactics, and weapon employment philosophies already coded in them. Variations in these tactics are controlled parametrically, and a well-structured questionnaire can help the analyst get this information. This questionnaire can be maintained by the pilots, or by observers during the test. It is also important to have the ability to replay the simulation. This is very valuable for ensuring that the tactical information gathered during the test is accurate. It is also necessary if completely new tactics have to be coded, since any written information gathered during the test is probably not adequate for implementation. Capturing threat tactics is equally important, and having some structure to the threat tactics employed in virtual simulation can aid in this process.

This paper is a declared work of the U.S. Government and is not subject to copyright protection in the United States.

The technical capabilities of the threat aircraft and weapons have an obvious effect on the effectiveness of a system, and the tactics used to employ it. But equally important are the capabilities of the pilots employing the threat system. It is very helpful if when implementing tactics in constructive simulation if the level of aggressiveness, sophistication, and coordination of the threat tactics used during the virtual simulation is well defined, and is recorded for each run.

INTRODUCTION

Man in the loop air to air engagement simulation is an important tool during the acquisition and development of aircraft, systems, and sub-systems. It can be used for requirements definition, concept of operations development, system effectiveness analysis, and analysis of alternatives. One of the major strengths of MIL simulation is that the operator is involved in the process directly, and can provide feedback to the developers and acquisition team. The problem with MIL analysis is that the number of runs that can be completed is very limited, and getting statistically significant results from this limited number of runs can be a problem. One way to augment the number of runs from a MIL simulation test is to translate the concepts of operation and tactics used or developed in the MIL simulation into a constructive simulation. Then you can use the constructive analysis to generate enough runs to validate the results and conclusions from the MIL event. This paper will highlight information that should be gathered from the MIL simulation and from the pilots who participate in the event, and show how to use that information in constructive analysis.

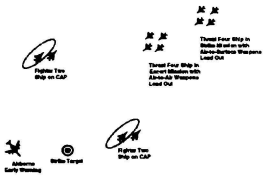
SCENARIOS & MISSIONS

There are many general categories of scenarios. A defensive counter air (DCA) scenario is a catch all for any time of defensive mission over home territory. A fighter sweep (FS) scenario is an offensive scenario where the

objective is to enter enemy territory and destroy as many enemy aircraft as possible. The Strike Escort (SE) mission is an air to surface mission with designated escort aircraft providing protection to strike aircraft whose primary mission is surface target attack. A Pure Self-Escort (PSE) mission is similar to the SE mission, but all of the aircraft are primarily surface attack, with enough air to air weapons to defend themselves on the mission. Cruise Missile Defense (CMD) is a variation of the DCA mission, where the objective is stopping a CM attack, while dealing any enemy aircraft sent to provide cover for the penetrating bombers of cruise missiles. High Value Asset Attack (HVAA) is an offensive air to air mission where the target is some specific airborne asset.

DEFENSIVE COUNTER AIR MISSION

Figure 1

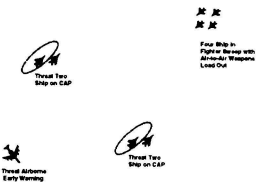


A defensive counter air (DCA) scenario is a catch all for any time of defensive mission over home territory. In Figure 1, two combat air patrol (CAP) points with two fighters each are defending an area against eight enemy aircraft. An airborne early warning aircraft is providing is providing command and control information, as well as situation awareness information on incoming aircraft via data link.. The eight enemy aircraft are divided by role. Four are escort aircraft, loaded with short range and medium range missiles, which are providing air cover to the attack aircraft. The surface attack aircraft are loaded with bombs and short range self-defense missiles only. The primary objective of the CAP aircraft in this scenario is to protect their target area by diverting or destroying the bombers, while avoiding being killed by the escort aircraft.

FIGHTER SWEEP MISSION

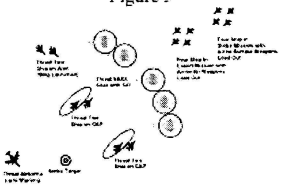
A fighter sweep (FS) scenario, as shown in Figure 2, is an offensive scenario where the objective is to enter enemy territory and destroy as many enemy fighter aircraft as possible. This type of mission would usually precede some sort of strike mission, the purpose being to soften up enemy air defenses. The primary objective of the CAP aircraft is this scenario is to survive, while destroying as many of the attacking FS aircraft as possible. The airborne early warning aircraft is providing is providing command and control information, as well as situation awareness information on incoming aircraft via data link.

Figure 2



STRIKE ESCORT MISSION

Figure 3

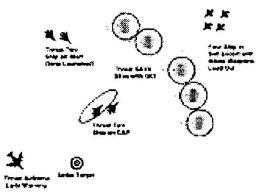


The Strike Escort (SE) mission is an air to surface mission with designated escort aircraft providing protection to strike aircraft whose primary mission is surface target attack. The eight SE aircraft in Figure 3 are divided by role. Four are escort aircraft, loaded with short range and medium range missiles, which are providing air cover to the attack aircraft. The surface attack aircraft are loaded with bombs and short range self-defense missiles only. The escort aircraft will attempt to divert or destroy the defending aircraft, thereby allowing the strike aircraft to fulfill their mission and destroy the target. Two combat air patrol (CAP) points with two fighters each are defending an area against enemy aircraft. Upon detection of invading aircraft, two more fighters are strip launched

into the scenario. An airborne early warning aircraft is providing is providing command and control information, as well as situation awareness information on incoming aircraft via data link.

SELF ESCORT MISSION

Figure 4



A Pure Self-Escort (PSE) mission is similar to the SE mission, but all of the aircraft are primarily surface attack, with enough air to air weapons to defend themselves on the mission, see Figure 4. This type of mission would be attempted when the enemy air threat is not perceived to be great enough to require escort aircraft. The surface attack aircraft are loaded with bombs and a limited number of medium and/or short-range self-defense missiles. The objective is to get through the enemy air defenses and attack the target, avoiding engagement if possible. Two combat air patrol (CAP) points with two fighters each are defending an area against enemy aircraft. Upon detection of invading aircraft, two more fighters are strip launched into the scenario. An airborne early warning aircraft is providing is providing command and control information, as well as situation awareness information on incoming aircraft via data link.

CRUISE MISSILE DEFENSE

Cruise Missile Defense (CMD) in Figure 5 is a variation of the DCA mission, where the objective is stopping a CM attack, while dealing any enemy aircraft sent to provide cover for the penetrating bombers of cruise missiles. This scenario is designed to represent a single air-to-air engagement in a first day of the war barrage attack with cruise missiles against a single target. Off-board surveillance and two CAP's of two fighters protect the target. Ten cruise missiles have been sent against a single target complex. The objective of the Blue CAP fighters is to protect the target.

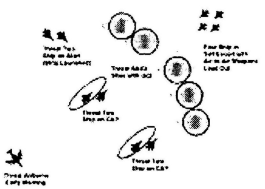
Figure 5



HIGH VALUE ASSET ATTACK MISSION

High Value Asset Attack (HVAA) is an offensive air to air mission where the target is some specific airborne asset. Figure 6 shows the four invading aircraft are loaded with short range and medium range missiles. Their objective is to avoid the defending fighter aircraft, destroy the airborne early warning aircraft, and return safely. Two combat air patrol (CAP) points with two fighters each are defending an area against enemy aircraft, and defending the airborne early warning aircraft as well. Upon detection of invading aircraft, two more fighters are strip launched into the scenario. The airborne early warning aircraft is providing is providing command and control information, as well as situation awareness information on incoming aircraft via data link.

Figure 6

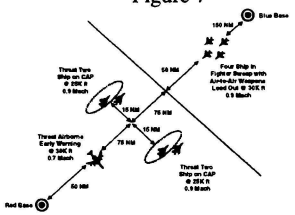


EXAMPLE SCENARIO

Figure 7 give some details of a possible fighter sweep scenario. The initial conditions shown might be typical of this type of engagement. Many of the initial conditions are fairly standard. Position, altitude, and speed of early warning aircraft and CAP aircraft to defend a designated area are pretty much standard based on system

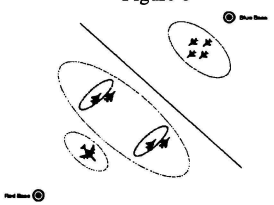
performance. Other factors may weigh in however. Initial separation between blue and red can be simply a function of detection range, a result of system performance. But if the detection range is long enough, the initial separation could be determined by the time required setting up tactics. The same can be said for CAP separation. It could be determined by system performance, such as data link range. It could be a function of the scenario, such as the width of the corridor to be defended. Or it could be a function of test objective, if a low threat density is desired with the two sets of CAP aircraft being engaged as two 4v2 engagements, rather than one 4v4.

Figure 7



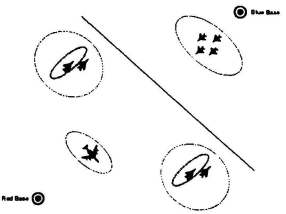
THREAT DENSITY & CLUSTERING LOGIC

Figure 8



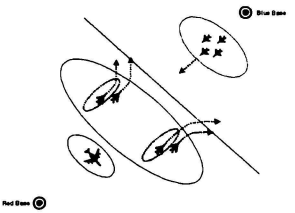
Clustering is the grouping of targets based on separation in range. If targets are close enough to be grouped into a single cluster, they are considered one entity for long range tactics. In this example the two CAP's are close enough to each other that the group of four fighters consider them one group, and will try to attack all four at the same time. The early warning aircraft is considered to be in a second cluster. On the other side, the group of four incoming fighters is very closely spaced, and will be attacked as a single group as well. This type of target clustering will result in a high threat density for the given scenario size, since all participants will be engaged at the same time. In Figure 8 the two CAP's are far apart that the group of four fighters consider them two groups, and will try to attack one CAP first, ignoring the other for the time being. The early warning aircraft is considered to be in a third cluster. On the other side, the group of four incoming fighters is very closely spaced, and will be attacked as a single group as well. This type of target clustering will result in a low threat density for the given scenario size, since the scenario will result in two 4v2 engagements, rather than one 4v4 engagement.

Figure 9



COORDINATION & INDEPENDENCE

Figure 10



There are a couple of other factors to consider when setting up a scenario, the level of coordination between the participants, and the level of independence each aircraft is allowed to exercise. In Figure 10, the blue side is flying a very simple straight in tactic, and is showing no independence. They have one lead aircraft, they will stay in formation, and if the lead aircraft is killed the rest will return home. Also if one aircraft is force to defend, all will defend together. The red side is flying a more highly coordinated tactic, a pincer, to gain tactical advantage.

However, they are still flying as a single group, will defend as a group, and will abort if lead is killed. Figure 11 shows a more independent tactic. Blue is now flying as two groups, each with a lead and a wingman. The tactics are coordinated between the groups, and they attempt to pincer the two red groups. Each group however is still flying as a single entity, and the wingman will abort if lead of his group is killed, and each lead and wingman pair will still defend as a group. On the red side, they are also flying as two groups, but are not flying a coordinated tactic as a four ship. Each lead and wingman pair is flying a different tactic, the left pair doing a simple point at target tactics. while the right group is trying to pincer the blue fighters. As with the blue aircraft, red will defend as a lead and wingman pair, and the wingman will abort if its lead aircraft is destroyed.

Figure 11

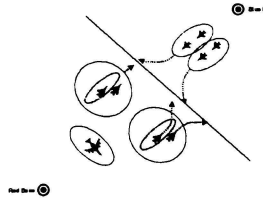
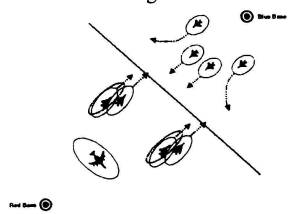


Figure 12



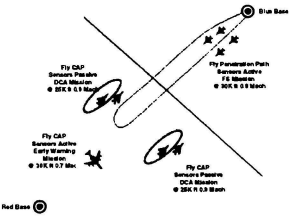
In Figure 12, both side are flying completely independent of each other. Defense and abort criteria are by individual aircraft, not lead and wingman pairings. Red side is flying a relative uncoordinated tactics, with each aircraft flying at its selected target. Blue is trying a more coordinated tactics by sending two aircraft up the middle, with the other two trying to pincer the threat aircraft.

MISSION PHASES

One way to implement tactics is to divide the mission up into "mission phases" based on range. The initial phase, neutral, is where an aircraft has not detected any threat aircraft, and is continuing with its initial tactic. This is usually following a defined penetration path, of flying a CAP orbit until something is detected. Once a target is detected, the aircraft will begin tactics to set up its attack. Early in the set up phase, the aircraft may simple continue it CAP orbit or penetration path. Once it has closed on the threat sufficiently, it will transition into tactics designed to gain positional advantage on the threat. As it continues to close, the aircraft will select its target, the move into position to launch a weapon at the target. This is the beginning of the attack phase. Once the target is attacked, some tactic will be required to support the weapon, while maintaining maximum advantage over the threat. After the weapon is autonomous, if the threat has not been destroyed, or another threat is still present, then some new close in tactic must be used, or the threat may be disengaged and set up for attack once some amount of separation has been achieved. If at any time during this sequence a missile is encountered, evasive action may be required, which may or may not take precedence over the current tactics.

NEUTRAL TACTICS

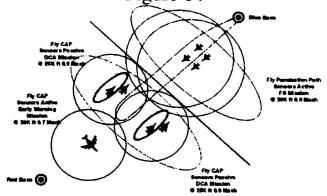
Figure 13



In the example FS scenario shown in Figure 13, the initial tactics are for red to remain on CAP at 25,000 feet and 0.9 mach with their sensors in passive mode. They are depending on airborne controlled intercept from the early warning aircraft to alert them to incoming threat aircraft. The blue aircraft are following a predetermined penetration path, flying at 30,000 feet and 0.9 mach. Since the have no ground or airborne surveillance support, they have all of their sensors active, in search of threat aircraft to attack.

EARLY SET UP TACTICS

Figure 14



Once targets have been detected, the aircraft transitions into the set up phase. In Figure 14 the dotted lines represent the clustering of the targets and the groups the opposing aircraft has put them into. The solid lines represent the aircraft groupings on their own side. Here both sides are grouped into to two ship formations, lead and wingman. All of the aircraft are fairly closely spaced, so both sides have grouped the threats in one cluster, so the engagement will be a high density 4v4. In this phase both sides will continue with their neutral tactics, maintaining the same speed and altitude. Sensor status will also remain the same, passive for the red fighters and active for the blue fighters. The size of the solid rings represents the range at which the aircraft will transition into the next tactics. Blue will change tactics before red in the example, as shown by the larger solid blue rings.

LATE SET UP TACTICS

Figure 15

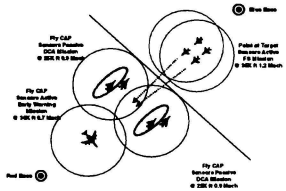
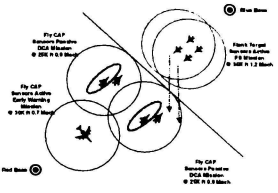
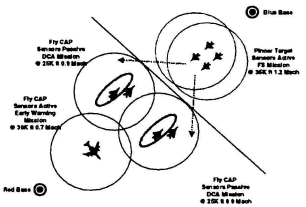


Figure 16



In Figure 15 blue side has closed sufficiently on the target that it has changed mission phase and moved on to a new tactic. Blue side has changed from follow penetration path to point at target as its primary tactic. Altitude has increased to 35,000 feet, and speed has increased to mach 1.2. Sensors are all in full active mode. Since all four red aircraft are still in one cluster, the attacking blue aircraft will point to the center of the cluster. Red side has yet to transition from the early set up phase, so its tactic remains fly CAP orbit. In Figure 16 blue side has closed sufficiently on the target that it has changed mission phase and moved on to a new tactic. Blue side has changed from follow penetration path to flank target as its primary tactic. Altitude has increased to 35,000 feet, and speed has increased to mach 1.2. Sensors are all in full active mode. This tactic tries to gain an advantage by keeping the known threats on one side of the formation, helping to make the engagement less dense. It also leaves blue in a better tactic situation after the initial merge than does the previous straight point at target tactic, since blue is less likely to end up surrounded. Since all four red aircraft are still in one cluster, the attacking blue aircraft will try to put all four red aircraft to one side. Red side has yet to transition from the early set up phase, so its tactic remains fly CAP orbit.

Figure 17



In Figure 17 blue side has closed sufficiently on the target

that it has changed mission phase and moved on to a new tactic. Blue side has changed from follow penetration path to pincer target as its primary tactic. Altitude has increased to 35,000 feet, and speed has increased to mach 1.2. Sensors are all in full active mode. This tactic tries to gain an advantage by surrounding the known threats, making the engagement denser. It will also leave red in a poor tactic situation after the initial merge, most likely ending up surrounded. Since all four red aircraft are still in one cluster, the attacking blue aircraft will try to bracket the four threat aircraft. Red side has yet to transition from the early set up phase, so its tactic remains fly CAP orbit.

In Figure 18 red side has closed enough on blue to come off CAP and begin setting up their attack against the incoming threats. Both pairs of CAP aircraft change tactics at about the same time, since the blue aircraft air coming in between the two CAP points. In this case both pairs of red aircraft fly a simple point at target tactic. Altitude is increased to 30,000 feet, and speed to 1.5 mach. Sensors remain in passive mode, with the intercept being flown from GCI and own ship passive sensor information. Blue has not yet close to the attack phase, and will continue to close on the red aircraft, using the lead target tactic.

Figure 18

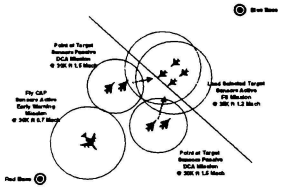


Figure 19 shows red side has closed enough on blue to come off CAP and begin setting up their attack against the incoming threats. Both pairs of CAP aircraft change tactics at about the same time, since the blue aircraft air coming in between the two CAP points. In this case the CAP aircraft effectively pincer the incoming threats by both pairs of red aircraft fly a flank maneuver in opposite directions to the target. Altitude is increased to 30,000 feet, and speed to 1.5 mach. Sensors remain in passive mode, with the intercept being flown from GCI and own ship passive sensor information. Blue has not yet close to the attack phase, and will continue to close on the red aircraft, using the lead target tactic. In Figure 20 red side has closed enough on blue to come off CAP and begin setting up their attack against the incoming threats. In this case both pairs of red aircraft try to flank the incoming aircraft to the same side. Altitude is increased to 30,000 feet, and speed to 1.5 mach. Sensors remain in passive mode, with the intercept being flown from GCI and own ship passive sensor information. For this tactic to be effective, one CAP must come off station earlier than the other, so they can join up before the incoming threats begin to engage the red aircraft as they cross their path. Blue has not yet close to the attack phase, and will continue to close on the red aircraft, using the lead target tactic.

Figure 20

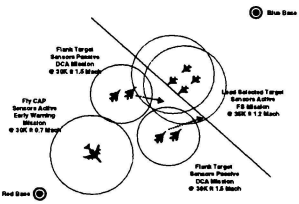


Figure 19

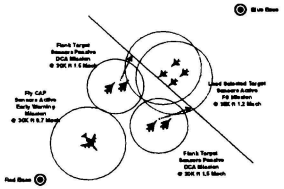


Figure 21

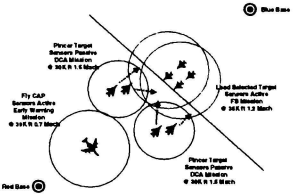


Figure 21 shows that red side has closed enough on blue to come off CAP and begin setting up their attack against the incoming threats. Both pairs of CAP aircraft change tactics at about the same time, since the blue aircraft air coming in between the two CAP points. In this case both pairs of red aircraft fly a pincer maneuver against the incoming threats. Altitude is increased to 30,000 feet, and speed to 1.5 mach. Sensors remain in passive mode, with the intercept being flown from GCI and own ship passive sensor information. The result of this is a low density attack by red, where all four incoming threats are engaged first by the two red aircraft moving directly towards the incoming aircraft, and later by the two red aircraft going to the outside of the incoming threats. Blue has not yet close to the attack phase, and will continue to close on the red aircraft, using the lead target tactic.

PRE-ATTACK TACTICS

Here the engagement has closed sufficiently to where each aircraft has selected an individual target, and is beginning to position itself for a shot. In this case the threat aircraft have been trying to bracket the incoming blue aircraft, so to set up the intercept geometry correctly, the red aircraft are being lead rather than pointed at directly. Sensors remain active, and speed and altitude remain 35,000 feet, and mach 1.2.

Figure 22

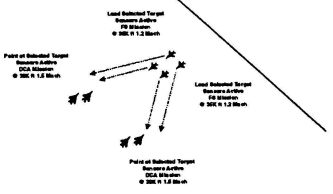
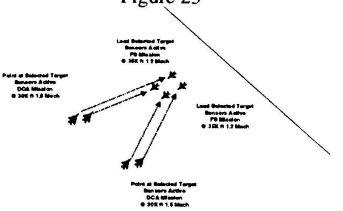


Figure 23



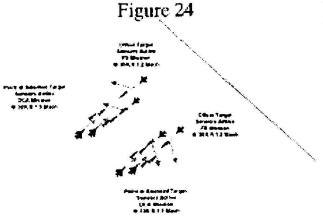
In Figure 23 the engagement has closed sufficiently to where each red aircraft has selected an individual target, and is beginning to position itself for a shot. In this case the threat aircraft have been trying to come between the red aircraft, and each red aircraft is pointing directly at its target in the threat group. Sensors are changed to active mode to provide higher track quality for weapon delivery, and speed and altitude remain 30,000 feet, and mach 1.5.

SHOT DOCTRINE

To determine when to release a weapon, missile kinematics and effectiveness must be considered. Missile range kinematics are a function of launch aircraft altitude and speed, target altitude and speed, and relative launcher to target range and geometry. Some commonly used terms to describe missile launch range are R_{max} and R_{ne}. R_{max} is the maximum range of the missile if the target does not maneuver. R_{ne} is the maximum range of the missile where no matter what the target does it cannot get away from the missile. Another commonly used launch range is 80% of R_{max}. Another thing to consider is how effective your weapons are at destroying your intended target. If the weapons are very effective, the shot doctrine might be shoot, wait and see if the target is destroyed, then shoot again if needed (shoot-look-shoot). If the weapons are not very effective, the shot doctrine might be to shoot two missiles one right after the other, see what happens to the target, the shoot again if need (shoot-shoot-look).

ATTACK TACTICS

Figure 24

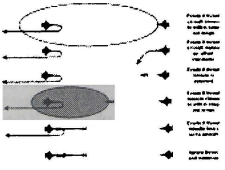


The attack phase shown in Figure 24 is the launch and missile support phase. Blue has taken a single shot at each threat aircraft. After the shot the offset from their target such that the missile can still be supported, yet maximum separation between the blue aircraft and the target can be maintained. The red aircraft are using a shoot-shoot-look doctrine, and are continuing straight toward the target after launching their missiles.

DEFENSIVE TACTICS

DEFENSIVE TACTICS

Figure 25



At any time during the mission, and aircraft may become aware that a weapon has been launched against it. How offensive or defensive an aircraft's reaction to a missile is going to have an important effect on the result of the engagement. Figure 25 illustrates some examples. A very defensive tactic could be to react to any threat aircraft that closes within a certain range. A large, slow aircraft with no offensive weapons might have to react this conservatively. A slightly more aggressive tactic is to observe the threat aircraft, and only react if there is some indication that it may have launched a weapon, such as a change in relative offset. More offensive missile reaction tactics may wait until a missile has been observed, observed within some range of own ship, is directed at own ship, or missiles can be ignored altogether. When defending against a weapon, there are a number of things to consider. Obviously the primary concern is evading the weapon that is attacking the aircraft. But speed, altitude, and relative geometry to threats in the area are also important. If you overreact to a threat weapon and lose all of your energy, you may end up in a worse situation than was required. Many long-range shots can be avoided by a simple offset maneuver, without ever losing the offensive geometry against the launching aircraft. Some weapons can be defeated by a beam maneuver, which may allow a more offensive re-engagement than a full turn and run maneuver. If the weapon has a lot of energy however, the defeat of that missile may require an all out maneuver, with little concern for what happens afterward, since if you don't beat that one nothing else much matters. Other factors that weigh in to missile reaction criteria are the state of an aircraft's own weapons. If the weapons are still in a state were support from the launching aircraft is required, the missile will probably be wasted if a defensive maneuver is made. Obviously if no weapon is in the air requiring support, an aircraft is free to react as other criteria dictate. But if a weapon is in flight, the launching aircraft can either wait until support is no longer required, or abandon the weapon and defend against the threat.

POST-ATTACK TACTICS

Figure 26



After the initial engagement has run its course as in Figure 26, the surviving aircraft must begin to set up the second attack. How this is done depend on the maneuverability of the aircraft, and the range of the weapons. If an aircraft is very maneuverable, but has shorter-range weapons, trying to merge into close-in combat after the initial attack may be the most effective thing to do. On the other hand, if the aircraft is not very maneuverable, but it has effective long-range weapons, disengaging from the fight to gain separation from the

threats, then turning back in for a second attack may be most effective.

DISENGAGE / RE-ENGAGE TACTICS

If after a disengage tactic a re-engage tactic is required, there must be criteria for when it is safe to re-enter the fight. If good information is available about the location of threat aircraft, then a re-engagement criterion based on range can be used. But many aircraft do not have that kind of information available. One way to get around that is to define a time after last indication of a threat behind you to wait to turn back into the fight, rejoin your penetration path, or return to your CAP point. If the disengagement was due to missile defense, then you could also turn back in after a period of time with no missile indication, regardless of threat aircraft status.

ABORT & SUCCESS CRITERIA

There must also be criteria for when the battle is over. If you are out of fuel or out of weapons, the mission must be aborted, and the aircraft return to base. If the aircraft is on a DCA type mission, there will be a limit on how far a threat aircraft will be chased from the area to be defended. In some cases, if an aircraft's wingman is destroyed that aircraft will return to base. If other cases, an aircraft may continue if its wingman is killed, but when the number of good guys drop below a certain level, it's probably time to call it a day. Another reason to abort is if the threat level is too high. Discretion is the better part of valor after all. Mission success is also a good reason to head for home. If the target is destroyed, or all of the enemy aircraft are destroyed or returning to base, your day is done as well.

DATA COLLECTION

During the MIL test, data will be collected to support the analysis objectives of the test. In addition to this information, other data will be desirable if a constructive simulation is to be set up for more analysis based on the MIL test results. It is useful to have the ability to play back the simulation graphically. This allows the analyst to quickly see how the engagement developed, and can help determine the criteria and tactics that need to be implemented in the constructive simulation to duplicate the engagement. Collecting information from the pilots about what the plan was going in is also very useful in implementing a scenario. Things to look for are initial set up tactics, weapon launch criteria, weapon defensive criteria, mission success and mission abort criteria, sensor usage, etc. The pilot debriefs after each run can also provide a wealth of information. Import information is how well did they stick to their initial plan, what did they do different and why, what did they do right and wrong, and what would they do if the situation came up again. Pilot perception is not always right however, so it is important to collect hard data to determine what really happened during the engagement. Sensor state, sensor detection performance, sensor ID performance, aircraft state, weapon launch range, etc are all important hard information for implementing a constructive simulation of tactics developed in a MIL simulation. When doing the constructive simulation runs, you will want to collect the same MOE and MOP information as was collected during the virtual runs. This allows a direct comparison of results, to see if the constructive analysis back up what the MIL simulation indicated. It is also good to collect the same performance data that was used to help develop the tactics implemented in the constructive simulation. You want the same sensor usage, sensor performance flight profiles, weapon launch ranges, etc that occurred in the virtual simulation.

CONCLUSIONS

Virtual simulation and constructive simulations used together can reduce costs and improve the quality of results. Virtual simulation is the best place for tactics development. Real pilots in the cockpit flying the scenarios provide the best source of concept of operations and tactics. It is difficult to evaluate and develop tactics without a real time operator in the loop. Virtual simulation can also quickly eliminate many parametric excursions in a test matrix. The operators can usually determine after a few runs what factors are driving the results, and which ones have little effect. Constructive simulation provides the number of repetitions required to eliminate anomalies in the results data. A single bad run in a virtual simulation can skew results, since so few runs are available. Constructive simulation is an excellent tool to provide the data to prove that such runs were in fact anomalies. Constructive simulation can also be used to look at further excursions in the test matrix that could not be included in the virtual test. The only caveat to that is that the excursions must be applicable to the developed and proven tactics, otherwise the results are probably not valid.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37278

AIAA-2000-4490

Air-to-Surface Virtual Simulation in Support of the Joint Strike Fighter's Joint Operational Requirements Document

T. E. Menke, S. T. Bishop, S. M. Rodgers, G. K. Hellmann, D. R. Mittlesteadt
United States Air Force/Aeronautical Systems Center
Wright-Patterson Air Force Base, Ohio 45433

ABSTRACT

Sweeping changes throughout the Department of Defense to improve efficiencies in acquisition processes have led to the creation of the simulation and analysis facility, or SIMAF, at Wright-Patterson AFB (WPAFB), OH. SIMAF is owned and operated by the Aeronautical Systems Center (ASC) in a substantial teaming partnership with Air Force Research Laboratories (AFRL).

A new modeling, simulation, and analysis federation has been formed through partnerships with engineering, analysis, and simulation offices at WPAFB. Within SIMAF this federation is conducting simulation and analysis for system and requirements trades in a way that is prototypical of the Simulation Based Acquisition vision. This capability enables collaborative teams of USAF decision-makers, war-fighters, and analysts to better and more quickly evaluate and develop advanced concepts and technologies for military applications. The analysis approach capitalizes upon the complementary strengths of constructive and virtual simulation methodologies. SIMAF provides cutting-edge visualization capabilities, computing power, and air-to-air and air-to-ground combat simulations.

The Joint Strike Fighter (JSF) Program Office has sponsored numerous virtual (manned) air-to-air and air-to-surface simulation events within SIMAF. The air-to-surface events are commonly labeled Virtual Strike Warfare Environment VSWE Events. The simulation scenario within each VSWE is a mission level environment encompassing Command and Control Off-Board Assets, Dynamic IADS, correlated Multi-Spectral DataBase (Visual, RF, IR), and two detailed piloted cockpits.

This paper will address the development and use of the VSWE environment for two VSWE events conducted in the summer and fall of 1998 for the Joint Strike Fighter Program. The two VSWE events (VSWE 2 and VSWE 3) targeted the Unit Interdiction (UI) and Choke Point Interdiction (CPI) JSF vignettes. This paper will discuss the UNCLASSIFIED level results of those events.

This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

INTRODUCTION

The Defense Department is in the midst of considerable reform in the conduct and management of its modeling and simulation activities. There are a large number of new initiatives and streamlining measures.

Jacques S. Gansler, the Under Secretary of Defense for Acquisition and Technology, noted in a 1998 address to the National Defense Industrial Association's Simulation Based Acquisition Workshop that one of the top challenges for the department was to modernize the military with drastically reduced cycle times and lower cost. Advances in the use of modeling and simulation technologies are widely recognized as key elements to meeting these challenges.

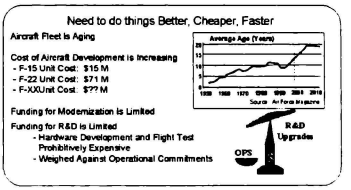


Figure 1 Aging Defense Systems and Limited Funding

The June 1995 4-star summit provided a New Vector for Air Force modeling and simulation. This vision endorsed by Air Force senior leadership calls for eliminating redundancy within the Air Force's inventory of Modeling and Simulation (M&S). The proposed consolidation will directly lower operations and maintenance costs associated with M&S by reducing multiple model upgrades and multiple, duplicative database maintenance. Many in the community thought that increasing the collaboration between existing users of models and simulation would speed the adoption of and migration toward common models and databases.

Aeronautical Systems Center (ASC), as an acquisition management organization, is tasked with the

development of Air Force systems. Inherent within this management duty are a number of tasks including:

- technical analysis of user requirements;
- technology research and development (R&D);
- dissemination of new technologies;
- applications of these technologies to Air Force requirements developers, systems prototype developers, and formal system program office (SPO) development organizations;
- developmental testing;
- transition of new systems to operational user testing and evaluation; and
- transition to operational logistics support.

In addition to the SPOs and planning offices within ASC, a number of offices within Air Force Research Laboratories are also hosted at WPAFB. There were opportunities for synergies among these ASC planning organizations, laboratories, and system program offices. Exploited correctly, these synergies would result in more effective and efficient development of systems that clearly satisfy the warfighter's needs.

This paper will provide the reader with a short overview of the facilities making up the WPAFB air combat MS&A federation, their capabilities, and examples of past and current programs. The concept of the federation, including the teaming of the virtual and constructive simulation experts, why it is necessary, and the benefits and problems associated with it will be discussed. Throughout the discussion, the emphasis will be placed on the federation's recently completed simulations in support of the Joint Strike Fighter (JSF) requirements definition. Examples will highlight the federation's role in Simulation Based Acquisition. This paper will focus the discussion of the virtual simulation as applied to the air-to-surface JSF missions conducted at WPAFB in 1998. The paper will conclude with a brief discussion on the future direction of the federation, simulation infrastructure, and the expansion of SBA through the use of distributive simulation.

BACKGROUND

To answer the challenges of reform and streamlining and to capitalize on the synergistic capabilities offered by the expertise present at WPAFB, ASC developed the simulation and analysis facility, or SIMAF. SIMAF is an important enabler for ASC to sustain itself as the Defense Department's air vehicle center of excellence for distributed modeling and simulation for systems development. To support this mission area, a Congressional insert of $7M was included in the FY97 Omnibus Appropriations Bill (S11936/HR4278/HR3610) for a "WPAFB Simulation Facility" as part of PE27601F, "USAF Wargaming and Simulation." The objective was to significantly reduce the deficiencies within ASC's M&S infrastructure. Also, this integrated facility was intended to be ASC's entree into the Joint Synthetic Battlespace.

One immediate need was a facility appropriate for conducting highly classified simulations. SIMAF would provide the environment for interdisciplinary virtual prototyping and testing. At this facility analysts, warfighters, and USAF decision-makers could collaborate to:

1) speed identification of promising technologies and concepts;
2) perform evaluations within a "system of systems" framework; and
3) add additional rigor to the acquisition process.

During the early development of SIMAF, the existing simulation facilities at WPAFB provided significant insight and perspective for SIMAF's definition, including required hardware, software, physical layout, and network connections. It was during this planning period that a new relationship between the AFRL Integration and Assessment branch's virtual simulation experts and the ASC Planning Directorate's analysts was formed. The members of these teams knew that the final objective was to use distributed simulation to identify, quantify, and develop promising technologies for the warfighter.

SIMAF was ideally situated among other existing simulation facilities at WPAFB to leverage existing capabilities and expertise. Within AFRL there has been the Air Vehicle Directorate's Integration and Assessment Branch (AFRL/VACD) and the Human Effectiveness Directorate's Pilot-Vehicle Integration Laboratory (AFRL/HECI). In ASC, the Engineering Directorate has had the Crew Systems Evaluation Facility (ASC/ENFC) in the same simulation complex since 1996. Also, the Planning Directorate (ASC/XR) has had analysis groups specializing in both constructive and virtual simulation. The vision for work in SIMAF then created an organizational bridge coupling ASC and AFRL resources. This arrangement produced the opportunity to foster a new air combat modeling, simulation, and analysis (MS&A) federation. This "new" federation was fortified through teaming partnerships that shared the facilities' resources, both computer and human.

WPAFB FEDERATION

Prior to this teaming with ASC's Planning Directorate, the technologies and concepts investigated in the Integration and Assessment Branch had never been

analyzed to determine their worth at the mission or campaign level. This changed starting with the recent VISTA Advanced Capabilities Study, which looked at aircraft agility through thrust vectoring, missile agility through off-bore sight target track capability, and a helmet-mounted display and targeting system. The Planning Directorate's analysts investigated what impact, if any, these virtual results had at the mission and campaign level through the use of constructive analysis. This activity fostered the idea that jointly, Integration and Assessment Branch virtual simulation engineers and Planning Directorate analysts could form a potent Air Combat MS&A team. As collaboration increased this idea became focused. Soon it was apparent that SIMAF was to be much more than a new "*physical*" facility. Rather, its unique worth was as a "*people*" facility. This team of virtual simulation scientists, engineers, and analysts would be the core of SIMAF's success.

ANALYSIS DESIGN

In the past, laboratory technologies and concepts had been largely evaluated in isolation—at the component or sub-system level. Rarely were technology assessments taken up to the mission level and rarer still to the campaign level. Of course, it is precisely at these higher levels that innovative technologies and new concepts should clearly demonstrate their military worth to secure development backing and funding.

Simulation Based Acquisition requires the application of the appropriate models to answer the questions being asked by the decision-makers. Simulations and exercises without analytic underpinnings designed to address these specific questions fail to realize the benefits that are sought under the Simulation Based Acquisition initiative. The results that come from the application of the considerable computational power available today can be quite convincing when based on sound principles and systems engineering methods. This requires a broad interdisciplinary team composed of members from the scientific and engineering fields germane to the systems being designed. In this application virtual and constructive analytical methods substantively complement each other.

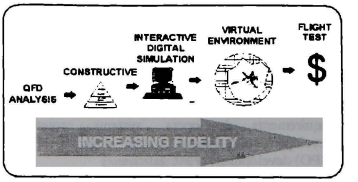


Figure 2 Simulation Spectrum

The main strength of constructive analysis is the ability to achieve statistical significance in the study results. These results are needed to validate the study's assumptions and to aggregate higher fidelity results to the next level of analysis. For example, constructive engagement-level results are aggregated for input to mission-level or campaign-level studies. Then, these constructive results are used as inputs and assumptions in virtual studies. Although figure 2 shows this as a sequential process, there is considerable iteration and results from the various studies can flow both up and down the spectrum. Another strength of constructive studies is to accomplish sensitivity analyses for given attributes of the system under test. These sensitivity and other constructive studies are well suited for screening efforts for the virtual studies.

The primary strength of "virtual" simulation is the increased realism regarding the application of tactics. This is especially the case in the reactions to free-flowing interactions of elements within the synthetic warfare environment. This increased realism provides excellent insight into pilot situational awareness, pilot-vehicle interface issues, decision making, and execution timeline issues. Situational awareness is further decomposed to encompass information display and processing issues, target and threat classification, discrimination, and identification issues, subjective issues, and more. These strengths are highly complementary in an integrated analysis plan and in the use of "design of experiments."

The benefit of placing pilots in the cockpit and collecting analytic data is substantial. The data is used to compare trends between the constructive and virtual efforts and to "tune" the constructive tactics and systems to both incorporate pilot comments and provide a bridge between the constructive "pilot-less" environment and the virtual "piloted" environment. The virtual analysis effort does not replace the constructive efforts; rather, it is complimentary in nature. Virtual efforts are both expensive and time

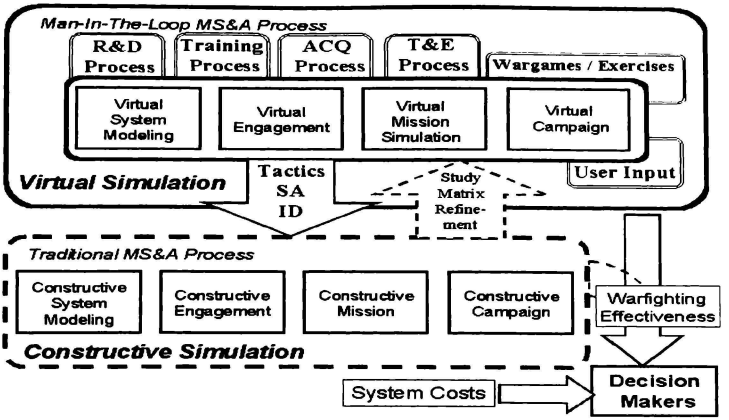


Figure 3. Constructive and Virtual Simulation are Complimentary

consuming. As such, a virtual effort can rarely satisfy the rigorous constraints of probabilistic analyses. Constructive modeling allows the analyst to perform a large number of runs in order to obtain statistical significance. By incorporating information and data from virtual simulation runs into the constructive process, the analyst can produce statistically significant results that are more representative of reality as a result of the presence of the pilot input.

In addition to mainstream analytical benefits of virtual simulation there are other useful effects from putting the operator "in the loop". One benefit that should be obvious but is at times overlooked by analytical purists is the convincing nature of participation in the events. Many decision-makers have a healthy skepticism concerning study results. In overcoming this skepticism there are few substitutes for getting the user involved in the details and nuances that are seen first-hand during participation.

Correlating constructive and virtual simulation is an iterative process. Pre-virtual constructive analysis can be used not only for statistical purposes but to screen the more expensive and time intensive virtual design of experiments. The goal of the prescreening effort is to identify system attributes worthy of more detailed piloted evaluation. This pre-screening effort feeds the Design of Experiment matrix for the virtual events. The analysis results of the virtual events are later correlated with the pre-virtual constructive analysis for significant trends in the various Measures of Effectiveness (MOEs). The results of the virtual simulation also support post-virtual constructive analysis. The virtual simulation's MOEs, Situational Awareness, Pilot Vehicle Interface (PVI), target Classification, target Recognition, target Discrimination (C/R/D), target Identification, and tactics are used by various engineering disciplines to correlate constructive analysis with piloted simulation results. The tactical refinement identified in the virtual events improves the statistical results obtained from the follow-on constructive analysis efforts. Often, virtual simulation provides unique and innovative insight in the employment of systems previously not conceived of by the engineering, test, and warfighter communities. Often particular attributes of interest become more important to objectively quantify following the virtual events. Post virtual constructive analysis begins with a Design of Experiments matrix based in large part on the analysis results from the virtual simulation event.

This analysis process is iterative in that pre-virtual, virtual, and post virtual analysis results focus the next modeling, simulation, and analysis cycle. Perceived Pilot Vehicle Interface deficiencies provide information and data to the weapon system contractors, human effectiveness, and the training communities.

THE ENGINEERING AND ANALYSIS TEAM

The engineering and analysis team comprises expertise from a large number of diverse disciplines including: project management; software development; hardware integration;, graphics development; aerodynamic performance, guidance, navigation, and control; threat intelligence modeling; database management; aircraft design; mission planning; and analysis. The core WPAFB team is comprised of approximately 35 people representing the system program offices, engineering, and laboratory organizations. This team is equally populated by both government and contractor members.

Event planning begins with software, hardware, and analysts meeting with the system program office and warfighters to address pressing questions in support of developing an operational requirements document. The analysis community scopes the attributes desired by the warfighters and system program office by iterating a development schedule with the software engineering and hardware team. The final product of this process is a detailed and highly integrated Design of Experiments based test matrix. The matrix is carefully screened to include blocking techniques to minimize individual pilot biasing effects on the resulting MOE trends while providing each pilot group with maximum exposure to each test attribute. The DOE test matrix is designed to reach a level 4 matrix resulting in primary test attributes that are independent.

The collaborative events done by this federation have become a prototype for SIMAF's SBA concept. The WPAFB air combat MS&A federation is uniquely qualified to provide the front end of the SBA approach and is positioning itself to complete the simulation based acquisition circle with the flight test and training communities.

SIMULATION AND ANALYSIS FACILITY

During the early development of SIMAF, the existing simulation facilities provided significant input into SIMAF definition, including required hardware, software, physical layout, and network connections. SIMAF would need presentation facilities for 100 people, floor space of over 40,000 square feet, segregated areas for classified work, connectivity, analyst workstations, and piloted simulators.

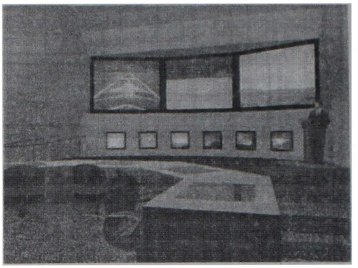


Figure 4 SIMAF Main Video Wall

Two-sided virtual wargaming-simulation and analysis can be done at the campaign level. SIMAF accommodates the Thunder model, in both real-time and compressed modes. Two "Team Areas" are provided; each "Team Area" accommodates 8 analysts, with the ability to isolate the two areas from each other. Team Areas have flexible computer processor and mass storage access, visual line of sight and direct audio contact with the video wall and simulation director areas. Each Team Area has a commander's station that includes a computer generated map table. The map table is used for monitoring analyst workstation displays, stealth viewing, moving map displays, and mission planning displays. Each map table has a capability for 2D stealth viewing, playback and recording.

Connectivity to numerous Wright-Patterson AFB assets has been demonstrated including the ASC Major Shared Resource Center (MSRC), AFRL/CFH, AFRL/AAS, and ASC/ENA. Connections to external assets (outside of Wright-Patterson AFB) have also been demonstrated. These include facilities at the Air Armament Center, Eglin AFB such as PRIMES and GWEF, and the "math lab". The Bennefield Anechoic Facility at Air Force Flight Test Center, Edwards AFB and the Air Combat Environment Test and Evaluation Facility (ACETEF) at the US Navy's Patuxent River Naval Air Station are planned locations for distributed simulations within the first year of operation. Other general wide-band connections exist and were used in 1999 to participate in EFX99 that included live, airborne test assets.

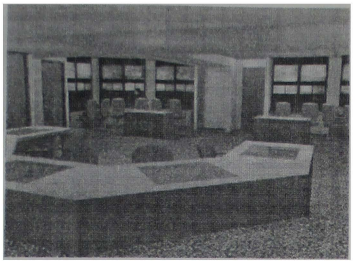


Figure 5 SIMAF Control and Work Stations

ASC, including all of Wright-Patterson Air Force Base, had no general purpose "special access" and "sensitive compartmented information" processing facility. Additionally, there had been limited capability for multiple, distributed real-time simulation coordination and control. SIMAF has resolved this shortfall in a big way. Up to 6 simultaneous yet isolated highly classified simulation events can be conducted in SIMAF. Any combination of connections between these events can also be realized as required by the various programs.

As the requirements for the facility were defined, the engineering and analysis team realized a "real-world" project would help rapidly identify and size the hardware and software resources that SIMAF would need. Through a series of meetings, the Joint Strike Fighter (JSF) program was identified as the ideal first program to be simulated in SIMAF and the first start-to-finish test of the engineering and analysis team's concept of how the air combat MS&A federation should function. Both air-to-air and air-to-ground simulation exercises were conducted for JSF.

PILOTED AIR-TO-SURFACE SIMULATION

Strike, SEAD, and other air-to-ground missions are performed in the Synthetic Warfare Collaborative Environment, or SWCE. SWCE consists of the fighter requirements evaluation demonstrator (FRED) cockpit and the joint interim mission model, or JIMM as the environment and interactions simulation engine. JIMM replaces the Simulation Warfare Environment Generator or SWEG in future VSWEs (see Figure 7). SWEG or JIMM is the main simulation model and incorporates a number of components including the Off-Board System Architecture, the Generic Composite Scenario (GCS), the Multi-Spectral DataBase (MsDB), and the Integrated Air Defense

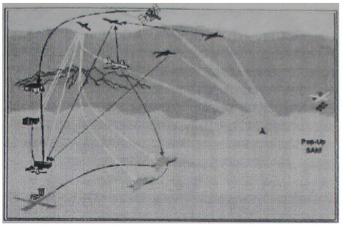


Figure 6 Mission Environment

System (IADS) (see Figures 6 and 7). JIMM interfaces with the pilot through FRED, and the visual Out-The-Window (OTW) displays. FRED interfaces with two Commercial Off-The-Shelf software (COTS) products for the detailed near physics level sensors, which interface with the MsDB. CAMBER is a Paradigm product that provides near physics level, real-time performance of a Synthetic Aperture Radar (SAR). SENSORVISION is a Vega product that also provides near physics level, real-time performance of a detailed Targeting Infrared (TIR) system. The OTW display including both terrain and the Heads Up Display (HUD) is provided by SUBRSCENE, a government product developed by Air Force Research Laboratories to provide high fidelity terrain imaging.

FRED is made up of several software modules, each supporting a different area of the simulation environment. The environment models manage the position orientation and velocity of the ownship, based on inputs from the flight control system. These models

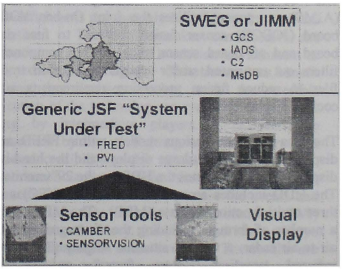


Figure 7. VSWE Components

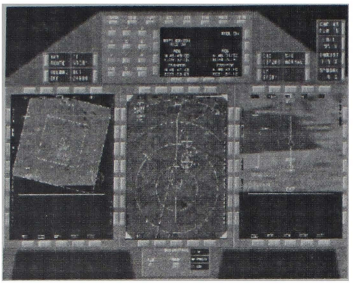


Figure 8 Generic Strike Fighter HDD

also manage the movement and weapons deployment of all non-ownship entities, and manage the flight of all weapons that have been launched by the ownship or by a threat. The sensor models provide data to simulate on-board sensors including an infrared search and track (IRST), electronic support measures (ESM), identification friend or foe (IFF), missile warning radar, a datalink, and targeting sensors.

The mission computer module serves as the core avionics system for FRED. It provides navigation data, and manages various components of the simulation based on pilot input including, route management, weapons stores management, tactical sensor management (e.g., sensor modes and fields-of-view), automatic modes management, defensive reaction module management, and sensor fusion management.

The advanced information management system (AIMS) component utilizes the Joint On-board/Off-board (JOBOB) sensor fusion software to fuse on-board and off-board sensor tracks. This component filters out unwanted sensor reports and fusion track files to reduce fusion processing requirements and cockpit display clutter.

The data from the various models within FRED are displayed on the head-down displays and the head-up display. FRED simulates the HDD on the 29" monitor. The HDD includes an up-front controller (UFC) and three multi-function displays (MFD). The MFDs hold a number of formats including the air-to-ground and air-to-air radar, a tactical situation display (TSD), an infrared targeting system format, an aircraft management system format, and electronic flight instruments. Figure 8 shows the radar format, tactical situation display, and the infrared targeting system on

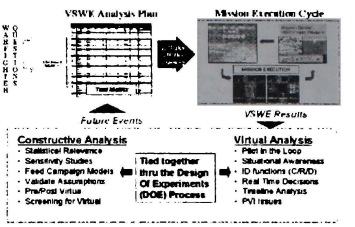


Figure 9 VSWE Analytic Process

the three MFDs. These displays are touch sensitive so that the buttons located around the perimeter of the MFDs provide pilot control similar to the actual cockpit.

The up-front controls are touch sensitive areas to simulate pushbuttons, a keypad area, and a data entry display. FRED also provides a simulated head-up display that is projected on a screen in front of the pilot station.

VIRTUAL SIMULATION EVENTS

The WPAFB JSF virtual simulation team executed it's first air-to-surface mission level event in August of 1998. The event supported the development of the third Joint Interim Requirements Document or JIRD-III, the last interim document before the release of the draft JORD. The second VSWE followed shortly thereafter in November of 1998.

The first air-to-surface event investigated the use of the Joint Strike Fighter (JSF) in a Unit Interdiction (UI) scenario with two JSFs executing a strike against armored columns in the GCS. The second event concentrated on SEAD and Time Critical Targets (TCTs).

Each mission required detailed planning with the Air Force Mission Simulation System (AFMSS). After the missions were planned, pilots downloaded their routes to SUBRSCENE allowing the pilots to become visually familiar with the routes and associated terrain prior to mission execution (see Figure 10). Following mission planning, the two pilots moved into the mission execution cell and fly two their planned mission twice. After completion of the mission execution cell, the pilots moved to the debriefing cell, which provided them the opportunity to review and debrief their previous mission. Each cell (mission

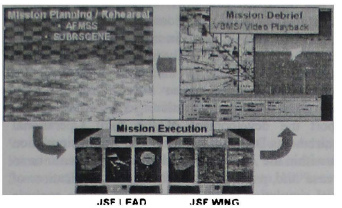


Figure 10 VSWE Execution Cycle

planning, mission execution, and mission debrief), contains a two pilot team. The team rotates every hour for 10 days until the test matrix is complete. Therefore in any given week, 6 pilots will continuously plan, execute, and debrief many sorties of the various system attributes in the test matrix for the selected mission scenarios. It is a detailed and rigorous schedule.

The test matrix investigated a number of radar, ir, and weapon system attributes for the selected missions. The test matrix is a Design Of Experiment (DOE) matrix of resolution four. A resolution four DOE allows for unconfounded primary attributes. The attributes selected for investigation during the VSWEs are determined based on the Operational Advisory Group's (OAG) requirement for data to support the development of the JORD. The OAG provided the analysis community with a large number of questions that required analytical underpinnings to support the development of the JORD. Each attribute selected for incorporation into the test matrix is correlated against the applicable OAG questions to ensure traceability within the test matrix. (See Figure 9)

Two days were spent on training during each week of the event. Pilots were briefed on the scenario, test matrix, analysis objectives, JSF modeled systems, threat systems, weapons, and pilot vehicle interface limitations. Training results were recorded to provide insight into the training effectiveness.

As previously mentioned, the conduct of the test allowed each side to plan their missions, execute those missions in the simulation, and debrief those missions using the test director (see Figure 10). Each simulation run was replayed in the debrief cell immediately after execution to allow pilots to review and critique their actions prior to completing a detailed pilot questionnaire. The pilot questionnaire was designed to capture subjective data not readily available within the objective simulation data. Hot washes are completed at the end of each day and an overall hotwash at the end of the week to allow the analysts and warfighters an opportunity to discuss the simulation results and provide final feedback on both the test results and any simulation pilot vehicle interface (PVI) issues.

VSWE RESULTS AND IMPACTS

The results from simulation events, conducted in support of the JSF JORD, have altered the quantity and quality of information available to the warfighter, analyst, and decisionmaker. The use of virtual simulation in the Joint Strike Fighter program has allowed the immersion of the warfighter in the proposed technology during the development of the requirements. This immersion significantly alters the warfighter's perspective in developing tactics to exploit the new system. Further, the warfighter gains insight into the requirement process not available by other more conventional approaches. The use of virtual simulation is essential in determining more esoteric but important attributes of the system such as situational awareness during a combat mission. Additionally, timeline analysis provides insight into the criticality of the arrival of off-board information to successfully survive or complete a mission. Virtual simulation also categorizes pilot perceptions and decisions regarding the classification, recognition, and discrimination of threats and/or targets during various missions and mission phases. Further, the use of virtual simulation is necessary for the employment of any subsystem requiring a real time decision from an informed pilot. These type of decisions are common for a pilot when employing nearly every sensor on a modern fighter.

The warfighter is not the only beneficiary from virtual simulation. The decisionmaker can observe the application of the system in a mission environment as employed by the warfighter. This tremendous benefit provides the decision maker with the credible application of the technology or system removed from editing by intermediate players. Further, the endorsement by the warfighting community provides additional credibility to the virtual simulation results. Other beneficiaries include the Weapon System Contractors (WSCs) who both witness the events and receive analysis data and pilot vehicle interface (PVI) comments from the Navy, Air Force, and Marine Corps pilots participating in these events. The results are used to further refine the contractor's future virtual simulation events ensuring a more robust, credible, and worthwhile simulation event. Additionally, many military organizations request data from these events to support ongoing research, current programs, analytic

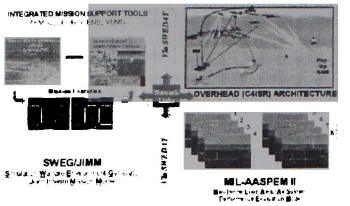


Figure 11. Future Federation Additions

development, or as support for simulation based acquisition (SBA) initiatives.

The Road Ahead

Future efforts are focused on addressing mission and campaign-level run-time issues with new meta-modeling approaches using response surface based techniques. This approach holds promise to increase the fidelity of the simulations without sacrificing run-time. Particularly interesting are the areas of high dimensionality due to interaction effects within the problem domains.

Additional theater assets are being added to the strike mission scenarios including developmental weapons and advanced command and control concepts. Legacy Models Interfaces are under development to improve the air-to-surface environment by leveraging existing capabilities such as the addition of the Man-In-the-Loop Air-to-Air System Performance Evaluation Model II (MIL-AASPEM-II or MIL2) (see Figure 11). The addition of a High Level Architecture (HLA) interface with the Digital Integrated Air Defense Network System (DIADS) provides a robust IADS useful to the Test and Evaluation (T&E) military community. Additionally, new approaches for modeling sensors are being explored. Further, efforts to ensure the simulation environment is suitable for test agencies include the generation of a multi-spectral database for the western test ranges.

SUMMARY

At WPAFB several diverse but related offices have joined to form a new modeling, simulation, and analysis federation. This federation is capitalizing on their human and computer resources to reinforce the application of solid systems engineering and analytical processes upon early acquisition efforts for military modernization.

The use of virtual simulation in support of the Joint Strike Fighter's JORD provides a prototype for both WPAFB and the Air Force in applying SBA concepts to DoD requirements analysis. To date, virtual simulation in support of the Joint Strike Fighter Program has resulted in an improvement in both quantity and quality of objective and subjective data to support the development of the Joint Operational Requirements Document. The use of virtual simulation both within the Air Force and Industry will continue to increase and expand in scope as the application of virtual simulation matures from requirements analysis to test and evaluation and training community needs.

The WPAFB virtual MS&A federation is uniquely qualified to provide the front end of the SBA approach and is pushing the simulation environment to complete the SBA circle with the flight test and training communities.

REFERENCES

1. Rodgers, S.M., et.al., Gansler, J. S. Under Secretary of Defense Acquisition and Technology, "Life-Cycle Modeling and Simulation A Key Element In Acquisition Reform," National Defense Industrial Association, Simulation-Based Acquisition Workshop, Orlando, Florida, March 17, 1998.

2. Moorman, T. S., Jr., General, USAF, Keynote speech to the Interservice/Industry Training System and Education Conference, Albuquerque, N.M., "Today's Technology, Tomorrow's Training," Nov. 14, 1995.

3. Sanders, P., Ph.D., Director, Defense Test, System Engineering and Evaluation, "Simulation Based Acquisition: An Effective, Affordable Mechanism for Fielding Complex Technologies," March 21, 1997.

4. Aeronautical Systems Center Simulation and Analysis Facility (SIMAF) Requirements Document, 1997

5. Menke, T. E., Briefing to the Defense Planning and Analysis Society, "The Use of Virtual Simulation in Support of the Joint Strike Fighter's Joint Operational Requirements Document (JORD): A practical Simulation Based Acquisition (SBA) example," April 14, 1999.

6. Pew, R. W. and Mavor, A. S., Eds, Modeling Human and Organizational Behavior: Applications to Military Simulations, Washington D.C., National Academy Press, 1998.

7. Joint Strike Fighter Virtual Strike Warfare Environment 3 Analysis Plan, Aeronautical systems Center, Virtual Simulation Group, 1998.

8. Joint Strike Fighter Aircrew System Advisory Panel Air-to-Air 2 Analysis Plan, Aeronautical Systems Center, Virtual Simulation Group, 1998.

9. Joint Strike Fighter Aircrew System Advisory Panel Air-to-Air 3 Analysis Plan, Aeronautical Systems Center, Virtual Simulation Group, 1999.

10. Rodgers, S.M., Hellmann, G.K., Richart, D.K., Mittlesteadt, D.R., Zeh, J.H., and Menke, T.E., "The "New" Air Combat Modeling, Simulation, and Analysis Federation at Wright-Patterson AFB, OH," May 1999.

11. Man-in-the-Loop Air-to-Air System Performance Evaluation Model (MIL-AASPEM) User Guide, Rev A, September 30, 1998.

A00-37279

AIAA Modeling and Simulation Technologies Conference and Exhibit 14-17 August 2000 Denver, CO

AIAA-2000-4491

JIMM: THE NEXT STEP FOR MISSION LEVEL SIMULATION MODELS

Joseph P. Nalepka
Air Force Research Laboratory
Wright-Patterson Air Force Base, Ohio 45433

Abstract

Two categories of simulation can be used to assess the performance and capabilities of one or more systems in a realistic environment. The first category, virtual simulation, can incorporate both humans and operational system hardware into the simulation environment. Virtual simulations are conducted at a real-time update interval and are typically used to train personnel on how to operate a specific system or to address safety concerns after an upgrade or modification has been made to a specific system. The second category of simulation, constructive simulation, does not typically incorporate humans or system hardware into the simulation environment and these simulations are usually operated at faster than real-time update intervals. Constructive simulations are typically used to fine tune virtual test matrices or for conducting trade studies. Because of their relatively low cost to set up and operate, constructive simulations are being utilized in many facets of system development. In particular, constructive simulations are becoming a critical asset in the Simulation Based Acquisition (SBA) Process. Because the SBA process requires realistic and detailed simulation conditions, it was necessary to develop a simulation tool that would provide a simulation environment acceptable for doing SBA analysis. This tool is called the Joint Interim Mission Model (JIMM). Through its generic nature of representing simulation entities, its data analysis capability, and its configuration management plan, JIMM can be used to support many different types of simulations as both a constructive and a virtual simulation tool.

This paper is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

Introduction

Simulations can be used to assess the performance and capabilities of one or more systems in a realistic environment. These simulations can be grouped into two categories: constructive or virtual. A virtual simulation is one in which a human and possibly operational system hardware interact within the simulation environment at a real-time update interval. The systems within this simulation environment are usually modeled at a very high level of detail and can have large computational requirements. Virtual simulations are mainly used to assess a system's performance, train personnel on how to operate a specific system, or to address safety concerns after an upgrade or modification has been made to a specific system.

In contrast, a constructive simulation is one in which a human and operational system hardware do not usually interact within the simulation environment. In addition, these simulations typically execute many times faster than real-time in order to process a large number of simulation scenarios in a relatively short period of time. The systems within this simulation environment are modeled at varying levels of detail. These systems can be represented as simple, point mass entities or as highly detailed, six degree-of-freedom representations, depending on the type of constructive simulation being utilized. Because these simulations execute at faster than real-time rates and are relatively inexpensive to set up and operate, they are used for conducting trade studies or fine-tuning a virtual simulation test matrix. In addition, these simulations can be used by engineers and analysts during the design and development of a future weapon system to quickly assess its operational effectiveness against current and future threats.

Because constructive simulations can be used for a variety of different analyses, they can be grouped into one of five categories:

- **Campaign Level** - Provides results concerning the overall outcome of a two-sided conflict involving many different systems that lasts for several days (Example: THUNDER).
- **Mission Level** - Evaluates the effectiveness and survivability of one group of systems against another (Examples: SUPPRESSOR, EADSIM).
- **Few on Few** - Evaluates the interactions between small numbers of systems during a specific scenario (Example: BRAWLER).
- **One on One** - Evaluates the interaction of one system against another under a very specific set of conditions (Examples: ESAMS, RADGUNS)
- **Engineering Level** - Evaluates the effects that a specific system or subsystem element has on another specific system or subsystem element (Examples: LCOM, FASTGEN).

The level of modeling in these simulations ranges from ones whose models are very detailed and narrow in scope (engineering level simulations) to those whose models are broad in scope and simplistically modeled (campaign level simulations).

One type of constructive simulation that has been receiving a lot of attention is the Mission Level Model (MLM). Because MLMs are useful for assessing a system's performance in a realistic, integrated, threat environment, they are being used as an instrument in the Simulation Based Acquisition (SBA) process. The SBA process is one in which the majority of the planning, the design, and the test of a weapon system or other product are done entirely on a computer. The end result of this process is a product that is produced faster, cheaper, and more reliably than its predecessors[1]. During the SBA process, the MLM is used to assess the benefits and drawbacks of a particular component that may be included on a future weapon system. The results of the MLM analysis provide design engineers and analysts with a "quick look" at which subsystem components will be the most effective before the weapon system is manufactured. These "smart" decisions early in the design process will prevent costly upgrades to correct weapon system deficiencies or problems after it is fielded.

Although there are many different types of MLMs available to the simulation community, none provide a complete and robust set of capabilities necessary to meet the very rigid analysis, test, and evaluation requirements of a SBA program. Consequently, a new MLM tool was created specifically for the SBA initiative. This tool, the Joint Interim Mission Model (JIMM), is a merger of the capabilities of the Suppressor MLM into the Simulated Warfare Environment Generator (SWEG) MLM.

The merging of two existing MLMs into one model was done for several reasons. First, one or both of these models have functional attributes that are required to meet the requirements for a SBA tool. Using these existing capabilities prevented a "reinventing of the wheel" when developing JIMM. Second, because software is being reused, the amount of time required to develop JIMM was reduced and thus it became available to the simulation community in a much shorter period of time. Third, because Suppressor and SWEG are widely used in the simulation community, their capabilities have been tested and validated under a wide variety of conditions. Fourth, this merger enables SWEG and Suppressor users to retain their existing MLM analysis tools and use their existing simulation databases with JIMM. For SWEG users, their databases are directly compatible with JIMM while the Suppressor databases must be translated into a JIMM compatible format via an automated database converter tool. Finally, and probably most important, because this model is derived from two widely used models, users of SWEG and Suppressor will be able to transition to JIMM with a minimal "learning curve" to overcome. Since its release in October 1999, several SWEG users have already transitioned to JIMM. The transition process for the Suppressor user community has occurred much more slowly, however. This is because not all of the Suppressor functionality has been implemented into JIMM.

The focus of this paper will be to discuss the evolution of JIMM, its architecture and features, and its model management.

Pedigree

As previously mentioned, JIMM is a merger of the Suppressor MLM and the SWEG MLM. Before proceeding with a discussion of JIMM, it is

necessary to briefly discuss the basic capabilities of the two MLMs from which JIMM was derived.

Suppressor

Suppressor is a non-real-time, constructive, MLM simulation tool owned and controlled by the Aeronautical Systems Center at Wright-Patterson Air Force Base, Ohio. It is used to evaluate weapons systems, electronic combat systems, or tactics in many-versus-many scenarios. These scenarios consist of a variety of military systems such as surface-to-air missile systems, antiaircraft artillery, fighter aircraft, bombers, anti-radiation missiles, electronic combat systems, and naval vessels. This MLM allows the user to define, at various levels of detail, the types of entities that will be included within a scenario as well as the way those entities will interact with each another. These entities and their interactions are developed using a model specific scripting language. If required, terrain and its affects on the entities and their related subsystems can also be added to a scenario. The Suppressor model is written in FORTRAN 77 and executes under the UNIX operating system.

SWEG

The SWEG MLM is both a real-time and non-real-time MLM simulation tool that is owned and controlled by the Air Combat Environment Test and Evaluation Facility (ACETEF) at Patuxent River Naval Air Station, Maryland. Like Suppressor, SWEG is also used to evaluate many different types of entities in many-versus-many scenarios. The modeling of these entities is done at various levels of detail using a model specific scripting language. In addition to modeling the specific makeup of an entity within SWEG, one can also use a set of generic functions to control an entity's actions within the scenario based on its "perception" of what is occurring within a scenario. In addition, the concepts of command chains and communication networks can also be defined within a scenario to allow an entity to "share" its perceptions with other entities within the scenario. This model is also equipped with a shared data interface that enables an entity external to the SWEG model to interact with the entities being controlled by SWEG. It is this feature that enables this MLM to run in a virtual environment with other simulations, simulation hardware, Command, Control, Communication, Computer and Intelligence (C^4I) systems, or piloted vehicles. The SWEG MLM is written in the C++ programming language and executes under the UNIX operating system.

JIMM

As previously mentioned, JIMM is the merger of the SWEG and Suppressor MLM tools. It is a data driven, event stepped, general purpose, conflict simulation system. An event driven model is one in which the interactions within a given scenario are updated when a new interaction occurs rather than at a specific increment of time. This methodology allows for very efficient and fast execution of the model because only those interactions that have changed are being processed.

As with SWEG and Suppressor, JIMM can simulate many different systems and war-gaming effects at various levels of fidelity. These include sensors (such as radio frequency, infrared, electro-optical, radar warning, sonar, and acoustic), communication systems, jammers, weapons, movers (such as tanks, planes, missiles, and trucks), signatures (infrared, radio frequency, and acoustic), and tactics. In order for the model to implement these systems and effects in a realistic fashion within a scenario, six generic functions were implemented within JIMM:

- **Move** - The process by which an object changes position and orientation within a scenario.
- **Shoot** - The transmission of matter or energy within a scenario with the intent of damaging or destroying a target.
- **Talk** - The cooperative exchange of information between entities or objects.
- **Sense** - The non-cooperative gathering of information about other entities or objects.
- **Disrupt** - The ability to interfere with the sending or communicating functions of an object.
- **Think** - Provides the capability to realistically model human behavior.

These six functions describe everything that objects within JIMM think about or perform in a scenario[2]. As a result, the user is able to create a very realistic representation of the entities and their interactions with other entities within a scenario.

This model is also equipped with a "shared memory" interface that allows it to interact with other simulations, simulation hardware, or vehicles. These interactions can occur either locally or through a distributed network using a High Level Architecture (HLA) interface. JIMM is capable of processing at both real-time and faster than real-time update intervals, thus enabling the use of the same model to conduct both constructive and virtual simulation experiments. As previously mentioned, a database conversion tool was created that will enable a Suppressor user to translate Suppressor databases into a database format that can be used by JIMM. This model is written in the C++ programming language and can be executed under the UNIX (LINUX, IRIX 5.3, IRIX 6.2, IRIX 6.5, Solaris 2.6) or Windows NT operating systems.

JIMM Database Structure

JIMM is composed of a series of databases, written in a model specific script language called the JIMM Conflict Language (JCL), that are linked or "compiled" together to build the scenario. The following are the databases within JIMM and their function[3]:

- **Language Database (LDB)** - Contains the simulation language and parsing instructions.
- **Icon Database (IDB)** - Contains the icons for the graphics portion of JIMM.
- **Type Database (TDB)** - Contains player structures, performance data, and tactics of the simulation entities.
- **Scenario Database (SDB)** - Identifies which player structures exist in the simulation, their command structures, their initial locations and their initial plans.
- **Runtime Database (RDB)** - Informs JIMM when to begin and end simulation execution, what type of data to record and what graphical display options to use.
- **Ground Database (GDB)** - Used for including terrain into the simulation.
- **Environment Database (EDB)** - Used for including terrain into the simulation.
- **Analysis Database (ADB)** - Used for extracting and analyzing data recorded by JIMM.
- **Configuration Database (CDB)** - Contains all of the functionality of the RDB as well as defines how to link external entities to JIMM.

Although it is not possible to discuss each of these databases in detail within the scope of this paper, it is possible to describe some of the specifics that are represented within these databases. The following will discuss how entities are modeled within a scenario and the type of analysis results one can expect after executing a scenario.

Entity Definitions

The basic component that is modeled within any JIMM scenario is the "player." A player represents something that exists in the real world. It can represent a single entity (such as an airplane), a collection of entities (such as a tank battalion), or part of an entity (such as a radar in a Command Center). The level of detail or fidelity used to represent the player is strictly up to the user. In addition, there is no limit on the number of players that can be defined for a JIMM scenario. Figure 1 shows a player definition written in the JCL. In this example, the player being defined is a generic Surface-to-Air Missile (SAM) site.

```
PLAYER-STRUCTURE sam_site
 TACTIC sam_tactics
 PLATFORM 1 sam_hq
  ELEMENT 11 sam_hq_ele DISCRETE QUANTITY: 1
   SUSCEPTIBILITY hq_sig
   THINKER 111 hq_thk      CAPABILITY hq_thk_data
   COMM-RCVR 112 comm_rx   CAPABILITY comm_rx_data
   COMM-XMTR 113 comm_tx   CAPABILITY comm_tx_data
 PLATFORM 2 sam_radar
  ELEMENT 21 sam_rdr_ele DISCRETE QUANTITY: 1
   SUSCEPTIBILITY element_sig
   SNR-RCVR 211 rdr_rx CAPABILITY rdr_rx_data
   SNR-XMTR 212 rdr_tx CAPABILITY rdr_tx_data
 PLATFORM 3 sam_weapon_site
  ELEMENT 31 sam_wpn_site_ele DISCRETE QUANTITY: 1
   SUSCEPTIBILITY wpn_site_sig
   WEAPON 311 missile CAPABILITY missile_data
    FUTURE-PLAYER missile_flyout DISCRETE 6 (COPIES)
 LINKAGES
  111 WITH 211    112 WITH 113
  211 WITH 212    211 WITH 311
END PLAYER-STRUCTURE
```

Figure 1 - Player Definition Example

Within the player structure are defined all of the components and characteristics for that player. The first of these is called the "platform." A platform gives the player a location, orientation, and the ability to move within a scenario. In other words, it is the part of the player that has a physical existence within JIMM. A player can have one or more platforms. From Figure 1, we see that the SAM site player is composed of three platforms which represent a headquarters building, radar site, and weapon launch site. One

of the properties of a platform is that it can be defined as critical or non-critical. A critical platform is one in which if it is destroyed, the entire player is destroyed. A non-critical platform (default setting for a platform) is one in which if it is destroyed, the player is still operational. If all platforms within a player are destroyed, the player is also destroyed[4].

To further define the player, each platform is comprised of one or more "elements." An element is the part of the platform that enables it to be detected or destroyed by other entities within a scenario. As with the case of a platform, an element has a criticality associated with it. If an element is "Discrete," there is a limit on the number of times it can be destroyed. In Figure 1, all elements can be destroyed a total of one time. This quantity, however, can be any value greater than or equal to one. If all discrete instances of an element are destroyed, the corresponding platform is destroyed as well. In addition, an element can be defined as "Continuous" which means it can be hit but never destroyed during the course of a scenario. The ability to detect an element is based on its "susceptibility." A susceptibility is a user defined quantity that defines the ability of a sensor system to detect a particular element[4]. Figure 2 shows an example susceptibility signature table for the headquarters building element of the SAM site player.

```
SUSCEPTIBILITY bq_sig
  EMITTIVE-EM-SIG-TABLE
    DIMENSION 1 FREQ (GHZ) 100.  300.  500  700
     DIMENSION 2 AZ (DEG) -180. 180.
      DIMENSION 3 EL (DEG) -90.  90.
       SIGNATURE (WATTS/STERADIAN) 1.E-10
     DIMENSION 2 AZ (DEG) 0. 180.
      DIMENSION 3 EL (DEG) -90.  90.
       SIGNATURE (WATTS/STERADIAN) 300.
     DIMENSION 2 AZ (DEG) -180. 180.
      DIMENSION 3 EL (DEG) -90.  90.
       SIGNATURE (WATTS/STERADIAN) 1.E-10
  END EMITTIVE-EM-SIG-TABLE
END SUSCEPTIBILITY
```

Figure 2 - Susceptibility Table Example

Within each element of a platform, the user can define various "systems." A system is the mechanism by which a player can accomplish various tasks such as detecting other entities, sending messages, thinking about a plan or maneuvering. There are eight different types of systems that are modeled in JIMM[4]:

- **Sensor Receivers (SNR-RCVR) and Transmitters (SNR-XMTR)** - Represents the capability to non-cooperatively obtain information about other players.
- **Communication Receivers (COMM-RCVR) and Transmitters (COMM-XMTR)** - Represents the capability to cooperatively share information between other players.
- **Shooters (WEAPON)** - Represents the capability of causing damage to other players.
- **Thinkers (THINKER)** - Represents the capability of processing information and affecting the internal operation of a player.
- **Disruptors (DISRUPTOR)** - Represents the capability to interfere with another player's attempt to use its sensors, communications, or weapons.
- **Movers (MOVER)** - Represents the capability to change the location or orientation of a player.

Each of these system types is defined by a certain set of characteristics. Figure 3 shows the characteristics for the radar receiver associated with the SAM site player. Another feature of JIMM

```
CAPABILITY rdr_rcv_data
 MAX-PARALLEL-TRACKS 2 (NO-UNITS)
 ONE-M2-DETECT-RNG   100. (KM)
 HITS-TO-ESTABLISH-TRACK 3 (NO-UNITS)

 RCVR-FREQ-LIMITS
  LOWER-FREQ-LIMIT   .90 (GHZ)
  UPPER-FREQ-LIMIT  1.10 (GHZ)
END RCVR-FREQ-LIMITS

 SNR-DOPPLER-LIMITS
  MIN-DOPPLER    10.0 (M/SEC)
  MAX-DOPPLER  4420.0 (M/SEC)
END SNR-DOPPLER-LIMITS

 SENSING-MODE-RATES
  TRACK-SENSING-RATE     0.312 (1/SEC)
  GUIDANCE-SENSING-RATE  0.407 (1/SEC)
END SENSING-MODE-RATES

 ANTENNA-PATTERN
  DIMENSION 1 FREQ (GHz) 0.9  1.05  1.2  1.7
   DIMENSION 2 AZ (DEG) 0. 180.
    DIMENSION 3 EL (DEG) -90  90.
     GAIN (DB)  -20
   DIMENSION 2 AZ (DEG) 0. 180.
    DIMENSION 3 EL (DEG) -90  90.
     GAIN (DB)   20
   DIMENSION 2 AZ (DEG) 0. 180.
    DIMENSION 3 EL (DEG) -90  90.
     GAIN (DB)  -20.
END ANTENNA-PATTERN

 RNG-ALT-CAPABILITY
  DIMENSION 1 RNG (M) 0.0   150.E3
   MIN-ALT (M) MAX-ALT (M) 0.0  10.E3
 END RNG-ALT-CAPABILITY
END CAPABILITY
```

Figure 3 - Sensor Receiver System Example

systems is that the user can specify how they interact with one another. This internal command and control structure within a player is defined through the "linkages" data item. For the SAM site player, the command and control structure is defined as follows:

- The radar sensor receiver is linked with the radar sensor transmitter
- The communication receiver is linked with the communication transmitter
- The thinker system is linked with the radar sensor receiver
- The shooter system is linked with the radar sensor receiver

These system pairings enable the user to define exactly how information within a player is to be disseminated. For example, the radar sensor receiver system is the system that is used to pass target information to the shooter system in order to determine if a weapon launch should occur.

Thus far, everything that has been discussed about a player concerns its composition and how information is transferred back and forth within the player. What has yet to be discussed is the mechanism by which human behavior is modeled within the player at a reasonable level of fidelity. This mechanism within JIMM is called a "tactic." Tactics are used so that a player can react to the changing events within a scenario. Within JIMM there are four main tactical events[4]:

- **Resource Allocation** - Defines how a platform should allocate resources, such as missiles, based on the information it has about other entities within a scenario.
- **Contingency Plans** - Defines how a player should react to changing events in a scenario.
- **Evaluation Rates** - Defines the maximum rate at which a player will think about doing something.
- **Maneuver Profiles** - Defines how a platform should move within a scenario.

Tactics are the most powerful player data item because they define exactly how the player will behave within the scenario. Figure 4 shows an example tactic in which Resource Allocation is used to determine if the weapon launch site of the SAM site player should fire a weapon at a target.

Because tactics tend to be the most complicated of all of the JIMM player definitions, it is beyond the scope of this paper to go into them in any further detail. Further information concerning tactics can be found in Reference 4.

```
RESOURCE-ALLOCATION
 LETHAL-ENGAGE-FIRING-START fire_weapon
  TGT-TYPE bomber
   USE INPUT FOR FILTER 1
    WPN-TYPE missile
     2D-DIST < 85.75 (NM)
     REL-TGT-HDG < 141.23 (DEG) AND
     FIRING-NOW IS NO AND
     TRACKING-STATUS IS TRACKING AND
     ROUNDS-FIRED-SO-FAR NO-MORE-THAN 2 (ROUNDS) AND
     TIME-SINCE-LAST-INTERCEPT > 3. (SEC)
   USE FILTER 1 SELECTIONS FOR FILTER 2
    WPN-TYPE missile
     AVAILABLE-WPN-RESOURCE AT-LEAST 1 (ROUNDS)
      RE: ORDNANCE missile_flyout
   FROM FILTER 2 SELECTIONS
    CHOOSE-FROM missile
    WITH-ORDNANCE missile_flyout ROUNDS: 1
    WITH-ELEMENT ANYONE
     PICK-AT-MOST 1 NOW 1 TOTAL
 END LETHAL-ENGAGE-FIRING-START
END RESOURCE-ALLOCATION
```

Figure 4 - Tactic Definition Example

Model Execution and Analysis

The ability to extract and analyze data from a set of simulation experiments is critical when conducting a simulation. Within JIMM, the user has a great deal of flexibility in determining how data is to be collected and analyzed. The main mechanism for doing this is through "incidents." An incident is an event that occurs during a particular simulation run. Incidents fall into one of eight categories[5]:

- **Movement Incidents** - Events pertaining to a platform's movement within a scenario.
- **Sensor Incidents** - Events which affect the operation of sensor receivers or transmitters.
- **Communication Incidents** - Events which affect the operation of communication receivers or transmitters.
- **Disrupter Incidents** - Events which deal with interfering with a player's sensor system.
- **Assignment Incidents** - Events that determine if a target should be considered for a weapon firing.
- **Engagement Incidents** - Events that indicate a target is going to be engaged or fired upon.
- **Weapon Incidents** - Events that pertain to a weapon launch.
- **Thinking Incidents** - Events that pertain to a player's thought process.

A complete listing of all incidents can be found in Reference 5.

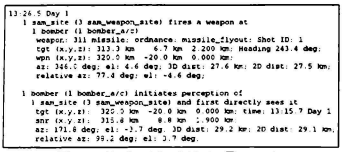


Figure 5 - Incident Data Example

Prior to running the scenario, the user can select which incidents should be recorded. An example of recorded incident data is shown in Figure 5. Another feature of the JIMM data analysis capability is that the user can filter the recorded data to further refine the data analysis effort. This is done using the JCL and a "sentence-like" filtering scheme that enables the user to extract very specific occurrences of one or more incidents. Figure 6 shows an example of how to filter the recorded data to find the number of times that one entity fired a weapon at another entity during a specific period of time of the simulation run. The resulting output is a histogram showing the results.

```
SENTENCE FILTER
  SITUATION: fires_weapon
    LISTING OFF                          $ Don't output incident details
    TIME-WINDOW: 1.0 7.0 (MIN)           $ Time Frame of Interest
    FREQUENCY INTERVAL 30.0 (SEC)        $ Resolution of Window
    ANYONE FIRES-A-WEAPON-AT ANYONE $ Incident to be analyzed
  END SITUATION

CORRESPONDING OUTPUT
  ==> Begin Situation: fires_weapon
  *--  MODEL MESSAGE 80 (INFORMATIVE): Summary of Incidents
       Frequency of Occurrence for Situations:
              FIRES-A-WEAPON-AT  13
         1:00.0 Day   1 1 *
         1:30.0 Day   1 1 *
         2:00.0 Day   1 2 **
         2:30.0 Day   1 1 *
         3:00.0 Day   1 1 *
         3:30.0 Day   1 0
         4:00.0 Day   1 0
         4:30.0 Day   1 2 **
         5:00.0 Day   1 2 **
         5:30.0 Day   1 1 *
         6:00.0 Day   1 0
         6:30.0 Day   1 1 *
         7:00.0 Day   1 1 *
    Scale: 1 asterix = 1 ( rounded up )
  ==> End Situation: fires_weapon
```

Figure 6 - Filtered Output and Histogram Example

Another tool that is available for analyzing JIMM data is a graphical package called OILSTOCK. This graphical tool is produced by the National Security Agency (NSA) at Ft. Meade, Maryland and it gives the user a visual picture of what occurred during a simulation run. In addition to graphically representing the various JIMM incidents, OILSTOCK also displays the location and movement path of entities within a JIMM scenario. This tool also has a playback capability that enables the user to review the events of a scenario run at various instances in time. Figure 7 shows an example OILSTOCK view of a scenario. The manipulation of the recorded JIMM data to OILSTOCK format is done through the use of the Analysis Database (ADB).

JIMM Configuration Management

To effectively coordinate changes and enhancements to JIMM, a Configuration Management Plan (CMP) has been established[7]. The agency overseeing JIMM development and the CMP is the JIMM Model Management Office located at the Electronic Systems Center (ESC), Hanscom Air Force Base, Massachusetts. This office is led by the Model Manager whose responsibilities are to manage changes to the model, assure that JIMM distributions are stable and fully functional and oversee implementation of the configuration management process. In addition, the Model Manager is the chairperson of JIMM Configuration Review Board (CRB) and Configuration Control Board (CCB). Working with the Model Manager is the JIMM Configuration Manager. This individual is responsible for reviewing, tracking, coordinating, and presenting all proposed changes to JIMM. Together, these two individuals are responsible for the overall integrity of the model.

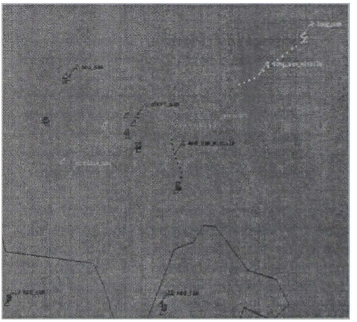


Figure 7 - OILSTOCK Example

In order to assure that the best interests of the entire JIMM user community are considered during the evolution of the model, a CRB and CCB were established. Both of these review boards consist of representatives from both

government and industry who utilize the model in a variety of different simulation applications. The purpose of both the CRB and the CCB is to assess the importance of changes or enhancements that will be made to the model. The CRB assessment is purely technical while the CCB takes into account programmatic information such as funding or the time to implement a proposed change. It is through these boards and their recommendations that changes to the JIMM model are prioritized and implemented.

Whenever a change is made to JIMM, the model goes through a rigorous verification and validation process. This is done to not only assure that the change or enhancement to the model was implemented correctly, but to also assure that the overall model performance has not been affected as a result of the change. In addition, backward compatibility is maintained between minor version releases.

Finally, in order to keep the user community informed about JIMM, user group meetings are held three times each year. These meetings are used as a forum to discuss the different ways that JIMM is being used as a simulation tool as well as to present and discuss the various changes and enhancements being made to, or being considered for, the model. It is also at these meetings that the CRB convenes to discuss the proposed enhancements and changes to the model. The CCB meeting is held soon after the CRB, usually as a telephonic meeting.

Conclusions

The Joint Interim Mission Model (JIMM) is being developed to support a wide range of virtual and constructive simulations. Through the generic functions of moving, shooting, talking, sensing, disrupting, and thinking, many different types of entities can be defined within JIMM that can behave and react in much the same way as a human. Also, because of the way scenario data is recorded in JIMM, analysts have a wide variety of ways to extract and process this data. In addition, because this model has been derived from two existing Mission Level Simulation Models, SWEG and Suppressor, transitioning these user communities to the use of JIMM is very easy and will enable these users to retain many of their existing databases and data analysis tools and techniques. As functionality and capability are added to this model, JIMM will become the first step towards implementing a versatile and robust mission level simulation standard into the modeling and simulation community.

References

1. Nalepka, J.P., Caudill, D.C., Mutschler, L., Buell, C., Garcia, L., O'Quinn, D., "Challenges in Integrating Commercial Off-The-Shelf (COTS) Software Into a Real-Time Simulation Environment," AIAA-99-4343, August 1999, Portland, Oregon.

2. Jones, Ross, "JIMM 1.1: Executive Overview," JIMM Class Notes, Fall, 1999.

3. Jones, Ross, "JIMM 1.1: Basic Training Course," JIMM Class Notes, Fall, 1999.

4. "JIMM 1.1 User's Guide - Volume 2," TRW Inc., 1999.

5. "JIMM 1.1 User's Guide - Volume 3," TRW Inc., 1999.

6. Kurker, Robert G., "Configuration Management Plan for the Joint Interim Mission Model (JIMM)," March 2000.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37280

AIAA 2000-4492

Man In The Loop/Hardware In The Loop Synthetic Battlespace Simulation

R. Perry, J. McDermott, M. Speed, W. Austin, M. Foster

Wright Patterson Air Force Base
Wright Patterson Air Force Base, OH

ABSTRACT

Man-In-The-Loop/Hardware-In-The-Loop Synthetic Battlespace Simulation

The ASC Simulation and Analysis Facility (SimAF) and the AFRL Integrated Demonstrations and Applications Laboratory (IDAL) have established a Simulation Based Acquisition partnership to conduct joint synthetic battlespace simulations fo support Joint Strike Fighter EMD. IDAL augments SimAF with the leading-edge simulation capabilities required to evolve next generation "system of systems" concepts/technologies where multiple sensors involving both airborne platforms and space assets are utilized to accomplish combat missions. These linked man-in-the-loop/hardware-in-the-loop real-time simulation capabilities produce virtual JSF aircrafts that fly virtual combat missions in high fidelity 2010 battlespaces. This strategic linking of spatially separated laboratory facilities reduces research costs, improves technology transition times and enables the collaborative research/technology integration required to develop leading-edge performance for the 21st century JSF fighter. This virtual battlespace testbed can develop, demonstrate and prototype offboard/onboard sensor capabilities that exploit information superiority to evolve revolutionary JSF capabilities. The IDAL/SimAF capability utilizes the Joint Interim Mission Model (JIMM) to generate the JSF synthetic battlespaces which provide a collaborative man-in-the-loop/hardware-in-the-loop development environment. This IDAL/SimAF partnership provides the Simulation Based Acquisition capability to develop better weapon system capabilities faster and cheaper.

MITL/HITL SYNTHETIC BATTLESPACE SIMULATION

I. INTRODUCTION

Modern test and evaluation laboratories face a unique challenge in order to provide relevant warfighter stimulation for Man-in-the-Loop (MITL) and Hardware-in-the-Loop (HITL) test configurations. The combination of multiple fidelity mission platforms and engineering level models must provide both fidelity and density to a simulated synthetic battlespace. Combining a synthetic battlespace with MITL and HITL requires a flexible laboratory configuration. The AFRL Sensors Directorate Integrated Demonstrations and Applications Laboratory (IDAL) comprise numerous HITL, MITL, signal simulators, and digital models that are used to represent the Warfighter's environment and threat systems. The IDAL provides a reconfigurable architecture to support warfighter demonstration/ analysis need.

IDAL's man/hardware-in-the-loop simulation (M/HITL) capability provides a cost-effective methodology for maturing advanced sensor technologies because the battlefield can be brought to the laboratory through synthetic battlespace simulation. IDAL's implementation of it's unique architecture feeds the goals/vision of the Simulation-Based Acquisition (SBA) process by assisting the warfighter/decision maker in:

(1) Identifying/resolving technology issues early in the acquisition cycle.

(2) Reducing time, risk, and resources associated with the acquisition process.

(3) Improving quality, military worth, and supportability while reducing development costs.

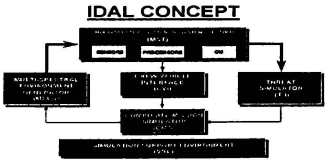


Figure 1: IDAL Architecture

IDAL is able to accomplish the goals of SBA by providing a controlled and repeatable test environment, which enables the advanced sensor technology to be subjected to multiple realistic combat situations. Figure 1 illustrates the IDAL architecture for supporting Warfighter demonstration/utility analysis. Referring to Figure 1, the Composite Mission Simulator (CMS) establishes and manages the overall engagement scenario characteristics during the Warfighter evaluation and then generates and transmits events to the Multi-Spectral Environment Generator (MSEG). The MSEG interprets the events and stimulates (either with real signals or digital representations) the system under test that resides in the Multi-spectral System of Systems Testbed (MST). The MST interacts with the Threat Systems (TS) by applying countermeasures and situational awareness to the aircrew through the Crew/Vehicle Interface (C/VI) instrumentation displays. The aircrew in the C/VI performs air vehicle responses based on the mission requirements and information presented on the situation displays. The Simulation and Support Environment (SSE) provides connectivity and simulator setup/analysis/visualization tools to support demonstrations.

The IDAL uses an incremental building block development methodology to support user demonstrations of military worth while increasing the laboratory's capabilities. This methodology builds on the "better-cheaper-faster" thrust of DoD. Figure 2 shows this methodology and the primary user/warfighter demonstrations that evolved IDAL's MHSBS. The Collaborative Enterprise Engineering (CEE) capability provides IDAL with the capability to distributively interact and share resources with spatially separated facilities and thereby add to and share IDAL MSSBS capability.

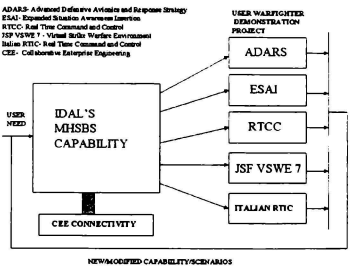


Figure 2: User/Warfighter Demonstrations

II. ADARS

The evolution of the IDAL's Man-In-The-Loop/Hardware-In-The-Loop Synthetic Battlespace Simulation (MHSBS) began with Advanced Defensive Avionics and Response Strategy (ADARS); an AFRL/SNZ 6.3 program. The ADARS program used actual data fusion algorithms such as Munkres association, Kalman Filter algorithms, and an IDAL based digital simulation environment. Figure 3 shows the IDAL ADARS simulation environment. The simulation environment consists of the Simulation of Air and Ground Engagement (SAGE), a mission/engagement level model that provides off-board TACELINT simulation and on-board ADARS player/platform data. The ADARS capabilities are the Multi-Sensor Correlater that correlates sensor data, Situation and Threat Assessment (level 2/3 data fusion algorithms) that assesses the situation, and Response Strategy that provides a pilot selectable response. The Response Strategy generates a Countermeasure (CM) program matrix of lethal threats vs. CM alternatives, which is a basis for constructing, evaluating, and scoring various CM response plans IDAL provided a MITL capability to demonstrate ADARS in a real time warfighter fusion utility demonstration using. The use of actual data fusion algorithms and appropriate equipment and simulator environment provided increased levels of confidence to resolve the value of off-board data for an airborne platform.

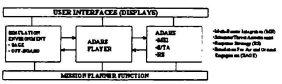


Figure 3: ADARS Digital Simulation Environment

III. JSF "GOODNESS OF OFFBOARD"

The next developmental milestone of IDAL's MHSBS evolved as part of the Joint Strike Fighter (SF) Program. The JSF will operate in a highly robust Command Control Communication and Computer Intelligence Surveillance and Reconnaissance (C4ISR) environment in the 2010 time frame and beyond. Projections for the comprehensive C4ISR architecture in this time frame suggest significant improvement in accuracy, timeliness, and amount of information available to the pilot with much improved latency by today's standards. IDAL's role for JSF analysis was to determine the survivability and lethality benefits of using "System of Systems" (SoS) on-board/off-board technologies and data for JSF.

The initial JSF analysis and technology demonstrations that IDAL provided were used as an input to the Joint Initial Requirements Document (JIRD) III.

The analysis and technology demonstrations IDAL produced for JSF were in two phases. The initial phase was constructive simulation that provided a series of computer-based model runs and analysis of the "Goodness of Off-board Data" for the JSF. The IDAL provided analysis with TACELINT data in order to provide a source of threat emitter situation awareness for JSF threat situation awareness and avoidance. The second phase allowed pilots from JSF's Operational Advisory Group (OAG) to fly through and interact with real and simulated off-board and on-board emitter data. In order to maintain compatibility with other JSF program studies, the IDAL used the Scenario Warfare Environment Generator (SWEG) V6.5.6 and Generic Composite Scenario (GCS) in order to maintain a consistent simulation environment. The JSF's "Goodness of Off-board" constructive simulation architecture was based upon the ADARS configuration as shown in Figure 4.

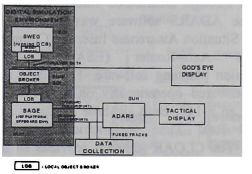


Figure 4. Constructive Simulation Architecture

The analysis used a networked set of computers running JSF-specific environment models and the ADARS tracker/correlater software and display. The environment consisted of:

SWEG/GCS — The preferred JSF mission level simulation model (now migrated to Joint Interim Mission Model (JIMM)) and is a Monte-Carlo based and event driven model. The GCS provides the JSF program with a threat laydown and set of JSF missions that encompassed the principal use of the JSF. SWEG is written in C++ and uses shared memory to interface with external simulations and simulators.

ADARS — Software provided the merging, correlation, and tracking of multiple source sensor information into a fused picture of the tactical situation.

SAGE — Provided mission/engagement level simulation for off-board TACELINT simulation and on-board JSF simulation. The SAGE translated between the mission level SWEG environment and the fidelity required to stimulate the ADARS data fusion algorithms.

Object Broker — IDAL-specific simulation interface backplane that continuously passes updated scenario/player data between SWEG and SAGE. The data is then shown on the "God's Eye" display.

The second phase of JSF analysis and technology demonstration IDAL performed was virtual simulation that allowed pilots from the Operational Advisory Group (OAG) to fly and react with real and simulated on-board/off-board ESM data. The two over-riding virtual simulation requirements were to demonstrate the goodness of SoS on-board/off-board sensors and data fusion in a JSF mission environment and to familiarize the JSF OAG pilots with real-time mission planning technologies. The approach used for the virtual simulation environment is shown in Figure 5. The changes from the constructive environment to the virtual environment were the substitution of the ADARS algorithms and integration with the MITL simulation through Distributed Interactive Simulation (DIS). The ADARS software was replaced by the Expanded Situation Awareness Insertion/Common Low Observable Auto-Router (ESAI/CLOAR) data fusion/mission planning software. The ESAI software was demonstrated operationally on a C-130 Combat Talon and an F-117. Both airborne platforms ran on a militarized Real Time Symmetric Multi-Processor (RTSMP). The CLOAR software provides an in-flight near real-time CM and maneuver recommendation for the pilot. At the conclusion of this user demonstration effort, IDAL had a robust sensor data fusion capability integrated with an airborne mission management system.

IV. REAL TIME COMMAND AND CONTROL

The next evolutionary enhancement of the IDAL's MHSBS added the ability to correlate information from National and Theater Intelligence broadcasts and from a UHF based Command and Control (C2) center. IDAL acquired the ability to receive operational off-board data for correlation with on-board sensor data. In 1999, IDAL used the correlater and data fusion algorithms

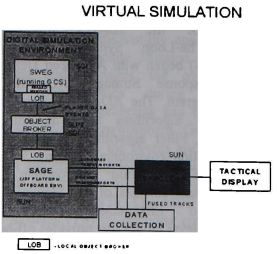


Figure 5. Virtual Simulation Architecture

from the updated ESAI program (MC-130H Combat Talon II, F-117A, and B-1B flight tests) to support a comprehensive laboratory demonstration. The laboratory demonstration goal was to develop/evolve the ESAI's advanced technology program by adding a Beyond Line of Sight (BLOS) and a fully implemented on-board/off-board data fusion capability. The National Systems data IDAL used was TRAP Data Dissemination System (TDDS) and the C2/mission update data was transmitted over a Quad-Zebra radio. The laboratory demonstration architecture is shown in Figure 6.

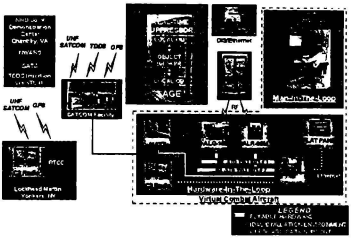


Figure 6. RTCC Architecture

The laboratory demonstration consisted of an Electronic Order of Battle (EOB) based on the South Western Asia (SWA)-2010X Multi-Service Force Deployment. The Air Force Standard Real-Time Suppressor model modeled the mission engagements. SAGE modeled the high fidelity end game engagements. IDAL used a DIS-based God's eye view, (Sky View), that presented the truth data within the battlefield and showed the virtual

combat aircraft in the overall/big picture within the unfolding scenario. A tactical display was the demonstration "show-piece". This display had "point and click" bezel buttons that provided the pilot mission waypoint data, sensor correlated threat data, new target information, target imagery, response strategy, and C2 information from the AOC. The IDAL MITL capability provides the virtual combat aircraft pilot interface and dynamic player interaction. The MSEG CEESIM (0.5-18 GHz RF simulator) was used to generate the representative threat emissions and stimulate the RWR and ECM systems. The Multi-Mission Advanced Tactical Terminal (MATT) receiver was used to provide BLOS information. The MATT received off-board TDDS National System information over a correlated EOB of the SWA-2010X scenario. The C2 radio provided UHF band mission data from a spatially separated AOC. The C2 radio provided RTIC text, retasking, imagery, threat data updates, and weather data to the pilot while pop-up threat data was RTOC'ed back to the AOC. The ESAI data fusion processor provided enhanced situation awareness and response strategy on a flyable Concurrent 6U VME card form on an air-cooled chassis. The National Reconnaissance Office (NRO) Demonstration Center provided simulated/correlated off-board threat intelligence data to operational National Systems through two of their models. The National Wargaming System (NWARS) model uses empheris data to calculate overhead data (with added uncertainty) at a particular time and date from the IDAL provided SWA 2010X threat laydown. The Stand Alone Tencap Simulator (SATS) allowed specific TDDS data injection and a simulated playback of the off-board data generated by the live broadcasts. The live demonstration links are shown in Figure 7.

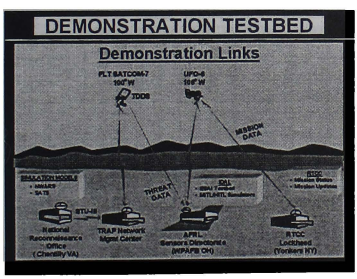


Figure 7. IDAL ESAI Demonstration Links

V. ITALIAN RTIC PHASE 0

The ESAI demonstration in March 1999 provides a tremendous increase in simulation capability for the IDAL facility. The next RTIC/RTOC demonstration planned in 4th quarter '00 is called the US/ITALY RTIC Phase 0. This demonstration enhances the IDAL MHSBS by adding a real-time airborne auto-router, Link-16 data link and three UHF DAMA compliant SATCOM radios. The scenario is a long-range strike aircraft that is entering the Balkan theater of operation en-route to a preplanned target that is re-tasked to strike a time critical target. The aircraft will maintain a simulated two-way connectivity with a C2 ground station throughout the mission. Within the scenario, the aircraft will receive updated threat information via TDDS, Link-16, SATCOM, and an on-board RWR. The aircraft will be capable of receiving, processing, and displaying threat data, mission re-tasking, and imagery data. The aircraft's mission management system will recognize when it has transitioned into or out of line-of-sight to the ground station and automatically switch between the DAMA SATCOM and Link-16 data transmission. Figure 8, shows the functional architecture of the program.

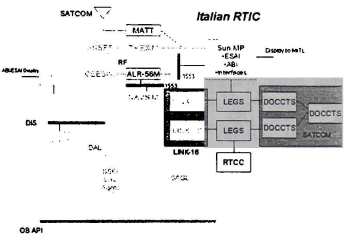


Figure 8. Italian RTIC Functional Architecture

The linkage and swapping/transmitting of all model data is established through the ethernet API. The SSE Link Agent (LA) provides a real-time executive, user interface, and visualization support to the link IDAL simulations and testbed. The Italian RTIC will utilize Real Time Suppressor (RTS) to provide control of the penetrating aircraft and extraction of engagement actions and results. SAGE generates end game high fidelity mission modeling and the Navigation Simulation (NAVSIM). The new additions to IDAL's synthetic battlespace include a Sun/Solaris-based CMMS that will now be commercially supportable, portable, and cheaper to maintain. Integrated with ESAI

is the Airborne Intelligence (ABI) capability that provides a consolidated picture of threat data using standardized symbology, provides a moving map, terrain display, and is currently fielded on AWACS. Another addition is the Real Time In Flight Planner (RTIP) that provides an optimized route replanner to increase aircraft survivability based on ownship RCS, overall threat environment, and mission objective. The Link-16 data link is provided using Via Sat LEGS and LINKS/S hardware. The Link-16 architecture provides the line of sight high bandwidth communication that allows all participants on the Link-16 network to receive C2 communication. Finally, a DOCCTS network is used for UHF SATCOM communication. The DOCCTS use current DoD SATCOM protocols and therefore is highly compatible with existing DoD communication networks.

VI. VSWE 7

The IDAL MHSBS building block evolution continues with a Simulation-Based Acquisition partnership with the Wright Patterson's ASC Simulation and Analysis Facility (SimAF). The partnership allows SimAF and IDAL to conduct joint synthetic battlespace simulations to support JSF requirement analysis and future EMD activity. IDAL augments SimAF with leading edge simulation capabilities that are required to evolve SoS concepts/technologies. The IDAL and SimAF linked M/HITL simulation capability matured as a result of the Virtual Strike Warfare Environment #7 (VSWE 7) exercise that was held in June 2000. VSWE 7 built from past VSWE (JSF simulation environment for requirement analysis) events which were used to develop the JSF Strike Warfare Collaborative Environment (SWCE). The SWCE is the high fidelity validated simulation environment and tool set JSF will provide for contractor EMD activity and will be used to support government Developmental and Operational Testing (DT/OT). All VSWE exercises support the development/evolution of the SWCE. SimAF's partnership with IDAL provides IDAL with an Asynchronous Transfer Mode (ATM) Permanent Virtual Connection (PVC) Defensive Research Engineering Network (DREN) connection. For the VSWE 7 activity and future simulation efforts, IDAL will augment SimAF with a HITL capability that is required to evolve next generation SoS concepts/ technologies. An SoS environment requires multiple airborne and space asset sensors that are utilized to accomplish combat missions. This strategic linkage of IDAL, SimAF, and other spatially separated facilities reduces research costs, improves technology transition times, and enables collaborative research/ technology integration required to develop leading edge performance for the JSF. The distributed battlespace will also demonstrate and prototype off-board/on-board sensor capability that exploits information superiority to evolve JSF capability.

The overall VSWE 7 exercise is a distributed four-ship of JSF cockpits that will be controlled through MITL C2 and opposed by MITL interceptors, SAMS, and C2. VSWE 7 will be the first distributed High Level Architecture (HLA) VSWE event. The primary sites were located at Wright Patterson AFB (IDAL and SimAF), NAS Pax River, and Edwards AFB. IDAL's role in supporting VSWE 7 was to integrate Real Time Joint Modeling and Simulation System (RTJMASS) technology and HITL (RWR and jammer) capability to support determination of the feasibility of using distributed T&E assets to test the JSF. The RTJMASS model was modified for the VSWE 7 exercise to accommodate:

(1) VSWE 7 C2 and responsiveness to the JSF Joint Interim Mission Model (JMM).

(2) Multiple target types and instances of targets.

(3) Ability to support sensor pointing angles.

(4) Ability to control multiple commanded missile assignments.

(5) Ability to represent target radar cross section for multiple targets.

An operational RWR and jammer will be used to stress the effects of latency for distributed test and evaluation. Within the VSWE 7 environment, the JIMM and RTJMASS radars (modeled at Pax River) drive a RF generator (CEESIM at IDAL) generate RF emissions representative of the JIMM radars to stimulate the system under test. IDAL's RF precision instrumentation measured/verified the correlation between the JIMM radars, RTJMASS missile fly-outs, and IDAL's RF simulator in terms of RF parameters and simulation execution timing.

The VSWE 7 architecture is shown in Figure 9. All distributed facilities run GPS synchronized environments via ATM networks routed to their respective Major Shared Resource Centers (MSRC) super computing facilities. All simulation sites ultimately receive connectivity by utilizing the DREN provided by the MSRC's.

VSWE 7 ARCHITECTURE

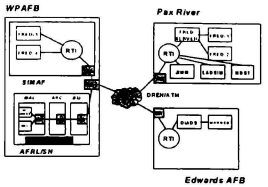


Figure 9. VSWE 7 Architecture

IDAL, using an IRIG time generator synchronized to GPS, will generate analysis data that examines latency and bandwidth issues. For latency, the delay within the whole network will be examined as well as the object HLA published times from Pax River through the JIMM model, and then the propagation of the same objects within the IDAL architecture. For bandwidth, sustained BW (6 mbps nominally) and loading effects (10 mbps peak) within the network will be examined.

VII. CURRENT CAPABILITY

At present the IDAL MHSBS capability is shown in Figure 10. Assets within the MST not previously mentioned include the ALR-56C, ALR-69, ALQ-165, and ALQ-135. By the third quarter in fiscal '00, a JTID's terminal and three DAMA compliant UHF radios will be added. Within the MSEG, we have three RF simulators (DEES, CEESIM, and EXCALSIM) with varying degrees of fidelity. The MSEG also includes the RISS (IR/EO missile warning simulator) and the EMCSIM (Electro-magnetic compatibility simulation).

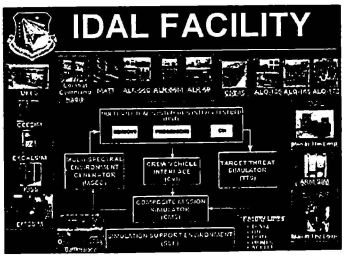


Figure 10. IDAL Capability

VII. CONCLUSION

IDAL's M/HITL simulation capability provides a cost-effective methodology for maturing advanced sensor technology because the battlefield can be brought to the laboratory through synthetic battlespace simulation. IDAL's incremental capability methodology implementation feeds the goals of SBA by providing a controlled test environment that enables advanced sensor technology to be subjected to multiple realistic combat situations. The sensor technology can be then subjected to multiple realistic combat situations. The sensor technology can in turn be demonstrated for the warfighter/decision maker so that he/she can:

(1) Identify/resolve technology issues.

(2) Reduce time, risk, and resources associated with the acquisition process.

(3) Improve quality, military worth, and supportability while reducing ownership cost.

IDAL provides a technology risk reduction capability that makes it possible to identify/resolve technology issues before flying or fielding the capability. Figure 11 depicts impact an R&D laboratory can have on weapon system affordability. IDAL's MHSBS is used to perform critical weapon system Warfighter/Decision Maker(s) (W/DM) proof of concept trades early in the development cycle. Risk reduction and testing permits the "knowledge curve of products and details" to be pushed to the left and the W/DM will have more flexibility to influence the detailed design and implementation of his/her acquisition system.

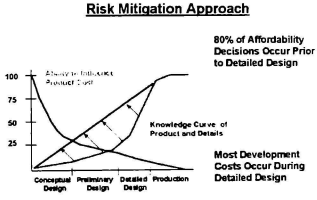


Figure 11. IDAL Risk Mitigation

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37281

AIAA-2000-4495

HIGH BANDWIDTH SCENE PROJECTOR DRIVE ELECTRONICS FOR IR EMITER ARRAYS

Matthew Price, InterSpace, Inc., P.O. Box 7206, Gaithersburg, MD 20898 (301) 527-0606. fax: (801) 720-8261
email: mprice@ieee.org.

ABSTRACT

Hardware-in-the-Loop simulators provide comprehensive ground test capability for end game testing of infrared (IR) interceptor concepts. Recent development has culminated in closed-loop testing involving large format resistive element projection arrays, 3D scene rendering systems, and real-time high fidelity phenomenology codes. Each of these components has been integrated into a real-time environment that allows simulators to perform dynamic closed-loop testing of IR interceptor systems or subsystems. This paper describes a unique array driver architecture that satisfies the bandwidth requirements of IR emitter arrays and is expandable in memory capacity, bandwidth and functionality required for planned future IR arrays. By taking advantage of recent advances in field programmable gate arrays (FPGAs) and memories, an IR emitter array driver has been designed with sufficient bandwidth and flexibility to meet the needs of current and future IR emitter arrays.

INTRODUCTION

IR scene simulation constitutes an important development tool for missile seeker and forward-looking infrared (FLIR) systems. By injecting real-time scenes representative of those obtained in the field into IR imaging systems, engineers can evaluate the end-to-end performance of the units. This process can be used to validate new designs, or to test production assemblies. In the future, this technology may also be integrated directly into imaging systems as a way to perform built-in test.

The bandwidth required to drive IR emitter arrays, used in hardware-in-the-loop simulators, has been steadily increasing. Advanced mega-pixel arrays are taxing the controllers, pixel buffers and processing electronics needed to drive these arrays. As array resolution and frame rates increase, the bandwidth required to drive these arrays increases exponentially. Table [1] shows some representative resistor arrays and their pixel bandwidth requirements for different programs. The CRISP program has a rather manageable bandwidth of 7.8 MHz, but the proposed AISP program requires a pixel bandwidth of over 200 MHz. At two bytes/pixel a data rate of 420 Mbytes/sec is required. Future arrays will require even higher bandwidth.

Program	Resolution (pixels)	Frame Rate (Hz)	Pixel Rate (MHz)
CRISP	512x512	30	7.8
NODDS	512x512	120	31.4
WISP	512x512	120	31.4
MSSP	512x512	120	31.4
DIRSP	1632x672	30	32.9
FIRSP	512x512	120	31.4
MIRSP	544x672	90	32.9
IRPSP	512x512	120	31.4
MIRAGE	512x512	200	52.4
AISP	1024x1024	200	209.7
future	2048x2048	400	1,678

Table [1]. Resistor Array Based Scene Projectors

This growing bandwidth requirement exceeds COTS driver solutions and even existing proprietary designs. Frame rates have increased to 200 Hz to reduce the need for test article synchronization and provide flickerless operation. Future IR emitter array temporal resolution is projected to increase to 400 Hz for Missile Warning Systems (MWS) and IR Counter Measures (IRCM). For low-bandwidth applications microprocessors or DSPs can generate pixel streams required for the arrays. At higher bandwidths this is no longer possible. Some manufacturers design custom hardwired circuits to drive these high-speed arrays, however this solution lacks flexibility. **What is needed is a flexible, high-speed driver capable of driving advanced resistor arrays.**

InterSpace, Inc. has designed a Video Display Processor (VDP) capable of meeting the unique bandwidth and functional requirements for advanced resistor emitter arrays.

Copyright © 2000 The American Institute of Aeronautics and Astronautics Inc. All rights reserved.

The VDP combines several features not found together in existing controllers. These features include:

- High Pixel Throughput – Over 800 million pixels per second
- Flexible Architecture – Reprogrammable FPGA based controller
- Compact Design – Single board PCB solution
- Advanced Pixel Processing Operations
- High-Capacity, High-Bandwidth Frame Buffer
- Modular Construction – Adaptable interfaces, expandable
- High Reliability – Self test, calibration
- Low Cost

The VDP incorporates local pixel processing operations thus relieving the host control simulator of some of the high bandwidth interface requirements. Functions include interpolation, scan rate conversion, windowing, calibration, and flat fielding (nonuniformity correction) among others.

SYSTEM ARCHITECTURE

In general, IR images may be separated into background, target and obscurations.[1] Conventional modeling systems have generated and combined these separate image sources into a single composite image for output to the emitter array driver electronics. As emitter bandwidth requirements have increased, this method has taxed the computational and I/O resources of the host processor and drive electronics. InterSpace, Inc. has developed a unique architecture that reduces the bandwidth requirements on the host and I/O channels by using separate asynchronous buffers for background, target(s) and obscurations. The images in these buffers are then combined in real-time to deliver the composite image to the emitter array. The emitter array drive electronics uses several high-speed, high capacity buffers and FPGA controllers as shown in Figure [1].

This architecture offers the flexibility of several interface modes. The Host can transfer data to the VDP in a conventional manner to deliver a composite IR image to the drive electronics. Alternately, the host processor can separately send target, background and overlay images to the drive electronics. The advantage of this later approach is that the host processing and I/O bandwidth is reduced by taking advantage of the different temporal and area resolution requirements of the images. For example, with many simulations the background image has lower priority than the target image(s) and thus does not need to be updated as quickly as the target image.[2]

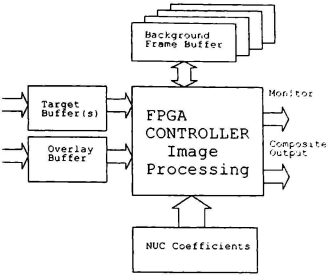


Figure [1]. System Buffers

Controller I/O Bandwidth

Figure [1] shows a central FPGA controller with multiple data ports. The controller bandwidth is summarized in Table [2].

DATA PORT	I/O Bandwidth (MBytes/s)
Target Buffer(s)	840
Overlay Buffer	840
NUC Coefficients Table	1680
Background Frame Buffer	840
Monitor Interface	420
Video Out	840

Table [2]. Controller I/O Bandwidth

The FPGA controller handles all the pixel processing and switching operations to support these data ports.

Target Buffers

The Host writes images of high-speed independent targets to the target buffers, and the VDP translates the images in real-time. Typically the target buffer is windowed down to a size smaller than the emitter array area. For example, with a 1024x1024 array, a target buffer window of 200x200 might be selected. This 200x200 target window could then be translated to any position in the 1024x1024 array area with sub-pixel resolution.

As way of illustration consider the following images. Below are two images; one of the background (from the Background Frame Buffer) and the other from a Target buffer.



Figure [2]. Background Image

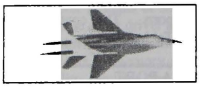


Figure [3]. Target Buffer Image

Figure [2] and Figure [3] are combined in real-time by the VDP to create the composite image shown in Figure [4]. Only the nonzero pixels in the Target Buffer image replace the background image.



Figure [4]. Composite Image of Background and Target Buffer Images

With this technique, the composite image is generated at high frame rates without increasing the Host drive requirements. For example, if the emitter array is driven at a 400 Hz output frame rate, the target buffer could be updated at a 60 Hz input frame rate, and the target *translated in position at the output frame rate* so that target movement occurs at a 400 Hz rate. The Host would only have to update the Target Buffer every 6-7 output frames. The target intensity can also be modulated every frame resulting in both movement and intensity modulation at the real-time rate (400 Hz).

Z-Buffer Support

The VDP supports multiple independent, overlapping Target Buffers. Pixels in each buffer contain attributes that specify target depth. This allows target images to overlap if on different z-planes—i.e., one target can pass behind another. The z-buffer is also used with background images so a target can pass in front of or behind terrain, such as when maneuvering around mountains or through a valley.

Host Interface

Typical IR simulations consist of fast moving target(s) on a slow moving background. A variety of target buffer interface options exist to allow the Host to select the method best suited to the application. The Target Buffers are configured as ping-pong buffers. Images are doubled buffered so as one image is written to the buffer, the previous image is read form the buffer. This allows the Host to update the Target Buffers asynchronously at a rate independent from the read rate. The Target Buffers support full bandwidth simultaneous read/write operations. Different interface configurations are summarized below.

Raster – The Host selects a rectangular Target Buffer window size and writes pixels to the buffer in a raster format. Typically the window size is smaller than the emitter array area size resulting in fewer pixels to write and a reduction in bandwidth. Further bandwidth reduction is achieved by moving the target buffer location between frame writes to reduce Host frame update rate. The window size is scalable up to full frame size. This interface option is compatible with existing video output ports such as the DDO2 port on Silicon Graphic workstations and PCs.

Vector – The Host can also address individual pixels in the Target Buffer using a vector write format. The data rate is content dependent since the Host only writes nonzero pixels to the Target Buffer. The Target Buffer controller executes an auto clear operation to the write buffer prior to Host update. The Target Buffer is double buffered as in the case of the raster format. This can lead to a Host data reduction for target images with fewer nonzero pixels, however additional overhead is required for addressing.

Silhouette Targets – For some applications it may be acceptable to render silhouette targets. This is similar to the vector approach described above except only the target image border pixels are written to the Target Buffer. The Target Buffer controller performs a real-time pixel fill operation of the target image on readout. The Host data rate is content dependent, however write bandwidth will be reduced since fewer pixels are written to the Target Buffer. An example of this operation is shown below. This method is particularly suitable for small, high-speed targets such as missiles and bullets.

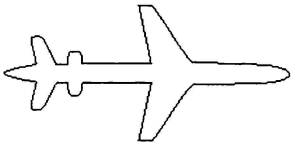


Figure [5]. Border Outline of Target Image

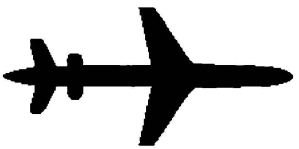


Figure [6]. Silhouette Target Image with Pixel-Fill

Vector write overhead is reduced since only address information is written to the buffer. Intensity values are written separately and used to modulate the target image at the output frame rate.

Hybrid – The Host Target Buffer interface can be configured for a combination of vector and raster writes or vector writes with different update rates. This is useful when rendering targets with rapidly moving components—such as helicopter rotor blades or missile exhaust plumes. For example, a helicopter image is written to the buffer using raster pixels, with the rotor blades superimposed on the target buffer by writing vector blade silhouette outlines with pixel fill.

Bandwidth Reduction

Separate target buffers offer a method to reduce the Host interface bandwidth requirements by decoupling target images from background and obscuration images. Full image bandwidth to the emitter array is achieved while using only a fraction of the corresponding Host bandwidth. This can allow an inexpensive desk-top system to drive a large format emitter array at full bandwidth. Table [3] summarizes typical bandwidth reductions that can be achieved with representative target images.

Host Interface (Raster/Vector)	Output Frame Rate (Hz)	Input Frame Rate (Hz)	Host Bandwidth (MByte/s)	Description: 1024^2 array MWS Application 16-bit pixels
R	400	400	839	Full BW
R	400	400	210	256^2 Target Full Update Rate
R	400	60	31	256^2 Target Translation & Intensity Mod
V	400	60	3	Target Pixels 10% of Target Widow
V	400	60	0.01	Silhouette Target Pixel Fill

Table [3]. Typical Host Bandwidth Reductions

Target Buffer Effects

The target buffer architecture also supports several additional features briefly described below.

Minimum Latency Mode – The read and write buffer pointers are set equal to allow target images to pass through the buffer without incurring a buffer delay. The target image is still stored in the buffer and is retrieved for dropout correction. This is useful for HWIL simulations to reduce feedback delay.

Graphics/Text/Color Pass-Through – It is useful to annotate targets with text parameters (such as descriptors, velocity, location, distance, etc) and/or graphics (such as cross-hairs, borders, vectors, etc.). This information is attributed separately from target image pixels to pass directly through to the Monitor Output Port without going to the emitter array. The attributed text and graphics follow the target image as it moves about the screen. This feature is useful for evaluation, test, development, demonstration and documentation purposes. In a similar manner target color can be written to the target buffer and passed directly to the Monitor Output Port with only the target luminance information going to the emitter array.

Collision Detect – Targets and background can overlap by assignment of different z-plane locations. If two targets (or a target and background) are assigned the same z-plane location then the FPGA controller can detect when these two images "collide." For example, a missile image is written to one Target Buffer and an airplane image is written to a second target Buffer on the *same* z-plane. When the missile pixels come in contact with the plane pixels a collision trigger is generated. This trigger can be fed back to the host for an interactive response, or can start/stop an image capture sequence. The same technique can be used to detect collision with a target and terrain. The collision detection occurs in real time on all image pixels.

Target Synchronization – Host updates of Target Buffers can occur asynchronously and are often at different frame rates from the output frame rate that drives the emitter array. It is necessary to synchronize the target image(s), background and obscuration images so precise timing relationships are maintained at the output. This is accomplished by attaching a frame time reference word to each buffer image update. The FPGA controller compares the buffer frame time reference with the real-time frame reference (system), and does not switch buffers (old-new) until the frame references match. In this manner the Host(s) can asynchronously update multiple input image buffers and maintain output synchronization.

Overlay Input Buffer

The Overlay Input Port is used to display IR images such as obscurations and atmospheric/weather effects. It supports transparency (pixel add) or z-buffer attributes (pixel replacement). In addition, text and graphics can be written to the Overlay Buffer and passed through to the Monitor Output Port, bypassing the emitter array.

The input to the Overlay Buffer comes from a video source. This source can be a standard analog signal (NTSC) from a video output device such as a VCR, DVD, IR Video Camera, or computer video card. The interface is also compatible with computer monitor drive signals such as VGA/EGA. The video signal is digitized by an analog-to-digital converter (ADC) to digital pixels which are written to the buffer. Alternately, the ADC can be bypassed if digital pixels are available from the source.

The Overlay Buffer supports asynchronous writes with a double buffered input for frame rate conversions. The digitizer clock (ADC) can be synchronously sampled or phase locked to a DDS clock from the video source (genlock). In this manner the Overlay image aspect ratio as well as frame rate can be adapted to the system clock.

Background/Capture Frame Buffer Port

The Background Frame Buffer serves two functions:

1. Provide Extended Scenario Playback Sequences.
2. Image Capture

The memory connects to the board using detachable modules (Figure [7]). Memory modules allow the user to configure the electronics for the memory capacity required for a particular application.

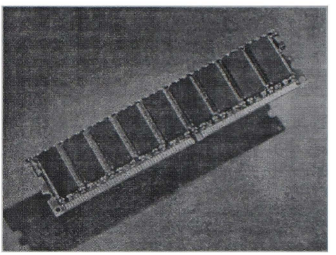


Figure [7]. Dual In-Line Memory Module

Scenario Playback Sequences

This high capacity memory can be used to store background sequences for playback in real-time. The playback duration depends on the playback frame rate and memory capacity. The playback frame rate may not be equal to the output frame rate since the background sequence frame rate may be slower than the target sequence frame rate. The difference between playback frame rate and output frame rate is referred to as the playback decimation rate. For example, if the output frame rate is 400 Hz and the playback frame rate is 100 Hz, the playback decimation rate is 4:1. Each background frame is repeated four times for each output frame. Playback decimation is suitable for most IR simulation applications since playback scenarios typically involve high-speed targets against lower speed backgrounds. Even though the background image may be moving at a slower frame rate, in this example, the target(s) are updated at the full output frame rate of 400 Hz. The playback rate can be selected for 1:1, however, if a full speed playback rate is required.

2D Scrolling Mode – Typical playback sequences contain a lot of redundancy from frame-to-frame. This redundancy can extend the playback period by using a scrolling mode of storage/playback. Instead of storing each individual frame in memory, the column and row increments are stored. As background frames are read from memory (during playback), the frames are shifted in the opposite direction of motion and the column/row increments appended to the frame. Playback is a smooth continuous motion. Compression rates up to 100x are possible. Typical playback durations are shown below in Table [4].

Capacity (MBytes)	Frame Size (16-Bit Pixels)	Decimation Rate	Output Frame Rate (Hz)	Scroll	Playback Duration (s)
8000	1024^2	1:1	400	no	10
8000	1024^2	4:1	400	no	40
8000	1024^2	1:1	400	yes	1000
8000	1024^2	4:1	400	yes	4000

Table [4]. Typical Playback Durations

Table [4] shows that real-time playback sequences of over an hour are possible using compression—even at output frame rates of 400 Hz.

Image Capture

The Image Capture Port can store full bandwidth image sequences in real-time. This port (and memory) is shared with the background memory. Image capture is useful for test, development, diagnostics and playback buffer initialization.

Capture Trigger Modes

The capture buffer requires a trigger to start image capture. There are several possible trigger modes. These include:

- External
  - Host sends trigger command to drive electronics
  - TTL – External signal generates trigger
- Internal – FPGA Controller Auto triggers when selected conditions are met
  - Target Position – Trigger occurs when target pixels intersect specified location/boundary
  - Frame Counter – Trigger when frame counter reaches specified value
  - Trigger occurs from z-plane intersection
  - Snapshot – Triggers occurs at periodic rate to store image snapshot(s)
  - Continuous – Retrigger when buffer fills

The trigger can also be advanced or delayed. Delayed trigger starts capture a specified number of frames *after* the trigger occurs. Advanced trigger starts capture a specified number of frames *before* trigger occurs. This is noncausal triggering made possible by continuously storing the image stream in a FIFO structure—when the trigger occurs the image history is in the buffer and retained.

Image Capture Source

The image capture source is selectable based on user requirements and applications. The source is configurable through FPGA initialization and can be changed on-the-fly by selection of the video capture source mux. Below are possible capture sources.

- Composite Image Prior to NUC (raw image)
- Composite Image after NUC
- Overlay Image
- Target Image(s)
- Video Input Port (Playback Initialization)

The capture source can also be windowed to a specified location on the screen. This window can be fixed or moveable during capture—to follow a target for example. Windowing capture data reduces stored data to an area of interest which reduces the data storage rate and thereby increases storage duration.

Buffer Management

The Background/Capture Buffer is a very deep buffer (up to 8 GBytes) that is used for both playback of background images and storage of image sequences. This buffer can be segmented so that different regions of the buffer are reserved for specific image sequences. For example, one half the buffer can be used for capture and the other half for playback sequences. The capture portion can likewise be divided into multiple sections for different capture sequences.

Simultaneous Capture-Playback

It is possible to capture image sequences to one portion of the buffer while they are read (played back) from another portion of the buffer (read-modify-write operation). This would typically be used to build initialization (playback) sequences or test/evaluation.

One application for this feature is to build scenario playback sequences. The capture buffer is first loaded with the background sequence *without* targets. The background images are then played back in an endless loop and a target is introduced to the sequence. When the playback loop reaches the end, it repeats with the target now in the sequence. Any number of additional targets can be added in this manner with the feedback loop.

Another application is diagnostics and verification. Intermittent errors tend to be difficult to detect and track down due to their transient nature. If a capture-playback loop is set up and run for an extended period, any transient pixel errors will "persist" in the capture buffer and be readily observable.

Monitor Output Port

The Monitor Output Port provides a standard video output source of the real-time image. Typically this output connects to a computer monitor as a means of viewing the video signal in real-time. Additionally, the video can be stored on a VCR or video capture PC board. In additional to displaying the composite background/target/overlay image, the monitor image can contain text, graphics and color that bypass the emitter array output image.

The Monitor Output Port connects to a dual-port, double-buffered memory that performs asynchronous frame rate conversions for the monitor interface. The system frame rate can be decimated to a suitable value for the monitor—typically 60-80 Hz. Also, progressive scan-to-interlace conversion is achieved for NTSC format. The FPGA controller writes pixels to the Monitor Output Port buffer at full bandwidth and reads then out at a reduced bandwidth for scan rate conversion. The digital pixels drive a Digital-to-Analog converter (DAC) where H&V synch and burst are added to generate a standard NTSC signal.

A pan and zoom feature is supported to extend the bandwidth limitations of video monitors and external storage devices. The output video can be windowed to show a smaller/higher-resolution portion of the emitter array image. This window can be positioned about the full memory image (pan) under user control or set to automatically follow a flight path. Image freeze and snapshot functions are also supported.

Separate user defined graphics can be added to the Monitor Output image. These can consist of cross-hairs, cursors, borders or icons. These graphics bypass the emitter array video image. The monitor port supports color output that can be used for pass-through purposes or to add false-color to IR images.

The monitor typically views the video image prior to NUC output. The FPGA can be reprogrammed to display any point in the acquisition/image-processing sequence for diagnostics or to suit a particular application

ALGORITHMS

The pixel processing algorithms are implemented by the FPGA in conjunction with external memory buffers. The FPGA offers several features making it ideal for this task.

- High System Clock Rate
- Wide I/O Width, Multiple Ports
- Multiple I/O Standards Supported
- Programmable
- Large Number of Gates
- Numerous Development Tools Available

This section describes some of the pixel processing algorithms targeted for inclusion in the FPGA.

Non-Uniformity Correction (NUC)

Each resistor emitter array has pixel-to-pixel non-uniformities specific to that particular array. The pixel non-uniformities are generally static in nature (fixed pattern noise) so can be measured in a calibration procedure and corrections made later during real-time operation. The resistors in the array also have a nonlinear response. Typically NUC both linearizes the resistor response and corrects individual pixel's non-uniformity. As emitter array size and frame rates have increased, NUC coefficient table capacity and bandwidth have correspondingly increased. The requirements for NUC are summarized below.

- Correct for Pixel-to-Pixel Non-Uniformity
- Correct for Non-Linearity (Linearize)
- Bandwidth: Real-Time (Pixel Output Bandwidth)
- Capacity: Full Frame (H x V Pixels)
- 32-Points Per Pixels
- Piecewise Linear Interpolation
- 16-Bit Pixels

Resistor Model

The typical resistor response of an emitter resistor[3] can be modeled by Equation [1]:

$$yr(x) := a \cdot \left(e^{\frac{k \cdot x}{x_{max}}} - 1 \right) \qquad [1]$$

where

yr(x)	=	resistor output
x	=	resistor input (DAC) value
a	=	curve shape parameter
k	=	normalization constant
x_{max}	=	range: 65,535 (16-bit pixels)

The graph of Equation [1] is shown in Figure [8].

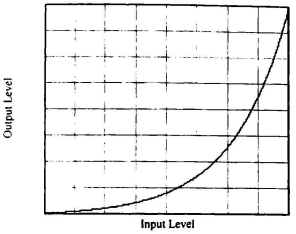


Figure [8]. Simulated Emitter Resistor Response

The NUC operation linearizes and normalizes the resistor response shown in Figure [8] for each individual pixel. This is accomplished by generating inverse curves for each resistor response curve. The ideal inverse curve for Figure [8] is shown in Figure [9].

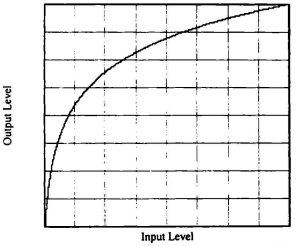


Figure [9]. Ideal Resistor Response Curve

Figure [10] combines Figure [8] and Figure [9] with the resultant ideal linearization curve (strait line).

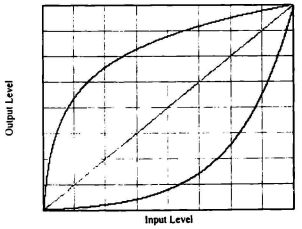


Figure [10]. Ideal Resultant Linearization Curve

Periodic Sampling

The ideal inverse curve uses 65,535 points to correct 16-bit pixels. It is impractical to use this many correction points for each pixel so an estimate of the inverse curve is made using a piece-wise linear approximation. For illustrative purposes, Figure [11] shows an 8-segment linear approximation to the inverse correction curve.

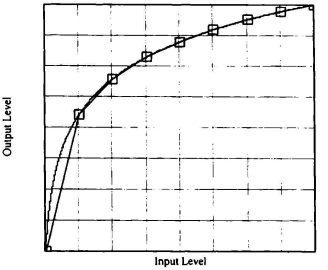


Figure [11]. 8-Segment Sample of Inverse Correction Curve

This 8-segment linear curve approximates the ideal correction curve and is used to correct the resistor output values shown in Figure [8] (in this example). The 8-segment curve requires storage of 16 coefficients; a slope and intercept value for each segment.

The composite graph showing the ideal (broken lines) and approximations is shown in Figure [12].

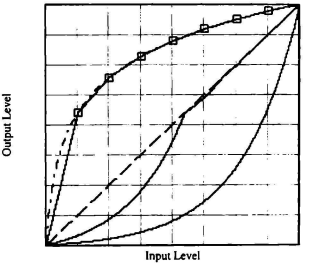


Figure [12]. Composite Ideal and Approximation

The 8-segment linear approximation leads to correction errors. These errors are characterized by coefficient errors and linearization errors. Linearization error is the difference between the ideal strait line and the corrected linear approximation (using interpolation), shown in Figure [13]. Coefficient error is the difference between the 8-segment linear approximation and the ideal inverse correction curve, shown in Figure [14].

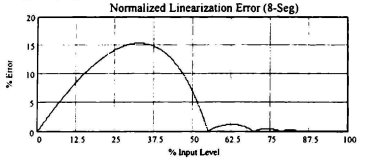


Figure [13]. 8-Seg Normalized Linearization Err

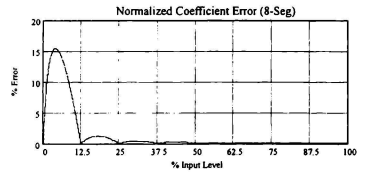


Figure [14]. 8-Segment Coefficient Error

Figure [13] and Figure [14] are normalized so the axis represents 100% full scale. The vertical axis shows the lower 20% of the graph for convenience. Notice the maximum error for both the linearization error and coefficient error is the same; about 15%. The errors are larger for low-level signals. This is apparent from the piece-wise linear approximation graph—the approximation is the poorest at low input values where the slope of the log curve is large and gets better for higher input values where the slope is smaller. Linearization "spreads-out" the coefficient error because the correction values transpose the x- and y-axis. This observation leads to a method of reducing both the maximum and mean correction errors by using non-periodic sampling.

Non-Periodic Sampling

By redistributing the sample values so that the correction curve is sampled at closer intervals where the slope is greater, and wider intervals where the slope is lower, the correction errors are reduced. The optimal sampling interval is exponentially distributed and shown in Figure [15].

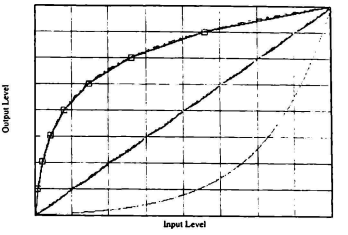


Figure [15]. Exponential Sampling

The same number of samples are taken so the number of coefficients remains the same. Figure [16] and Figure [17] show the linearization and coefficient errors for non-periodic sampling. The maximum error is reduced to less than 1% and the mean error about ½ %. The errors are distributed evenly throughout the input range.

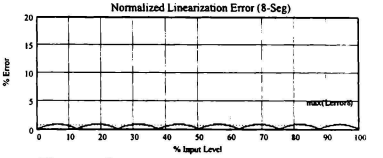


Figure [16]. Normalized Linearization Error

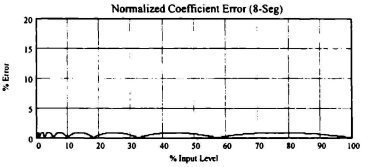


Figure [17]. Coefficient Error

The preceding graphs have shown the results for 8-segment linear interpolation. This was used for illustration since higher sample approximations results in errors so small they are difficult to show graphically. Table [5] summarizes the advantages gained through higher sample approximations.

Interpolation Seg	Mean Sampling Error %				bits
	Periodic		Exponential		
	Coefficient	Linearization	Coefficient	Linearization	Mean Bit Equivalent
8	1.4	5.7	0.57	0.57	7.5
16	0.5	2.6	0.14	0.14	9.5
32	0.1	1.0	0.04	0.04	11
64	0.04	0.3	0.009	0.009	14

Table [5]. Approximation Error Summary

With periodic sampling, as the number of coefficients (segments) are doubled, the maximum sampling error is reduced approximately by half. Periodic 64-segment interpolation results in a 1.9% maximum error. With exponential sampling, doubling the number of coefficients reduces the maximum error by about one-forth. Exponential 64-segment interpolation results in a 0.02% maximum error. The coefficient and linearization errors are equivalent for exponential sampling (max and mean). The mean bit equivalent is the equivalent number of bits corresponding to the mean error. Since 16-bit pixels are used, 16 sets the maximum desirable threshold for bit equivalent error. 64-segment linear interpolation results in a 14-bit linearization error using exponential sampling. This corresponds to the resolution of 14-bit DACs typically used for emitter array resistor drivers.

There are several different methods to implement NUC. Exponential sampling and linear interpolation results in an accuracy comparable to the resolution of the DACs used to drive resistor arrays. This method does not require a separate "compromise" linearization table as is used in other designs. By using a single coefficient table for both linearization and non-uniformity correction, each pixel can be corrected to the optimal uniformity value *and* linearization. The corrected output can be made monotonic over the operating range by selecting monotonic samples.

Coefficient Generation

The coefficient table stores two values (slope and intercept) for each sample point of the correction curve. What is referred to as 32-point interpolation would actually be 33 samples resulting in 32 piecewise linear segments and 64 coefficients/pixel. Correspondingly, 64-point interpolation results in 65 samples, 64 piecewise linear segments and 128 coefficients/pixel.

Pixels are corrected using the interpolation slope-intercept formula

$$yi(x) := M_m \cdot x + B_m \qquad [2]$$

where

M_m = Segment Slope Coefficient
B_m = Segment Intercept Coefficient
yi(x) = Interpolated value (output)
x = Input value
m = Coefficient table index

The relationship between coefficients and samples is given by

$$M_m := \frac{yc_{m+1} - yc_m}{xc_{m+1} - xc_m} \qquad [3]$$

$$B_m := \frac{xc_{m+1} \cdot yc_m - xc_m \cdot yc_{m+1}}{xc_{m+1} - xc_m} \qquad [4]$$

where

(xc_m, yc_m), (xc_{m+1}, yc_{m+1}) = 2 adjacent samples

Correction curves are acquired for each pixel during a time consuming calibration process. It is not necessary to reacquire the calibration curves if a different number or spacing of samples are used in the coefficient table. For example, if the correction curves were acquired using 32-point periodic sampling, a 64-point exponentially distributed coefficient table can be generated from the original samples. This is accomplished by first computing an exponential curve fit using the original samples. The exponential curve is then transposed to a continuous logarithmic curve to generate the correction curve. This continuous correction curve is then sampled and processed with Equations [3] and [4] to compute the coefficient table.

Saturation Arithmetic

Saturation arithmetic prevents MSB errors and allows overflow to occur in a gradual manner. An MSB error can occur when adding fixed point numbers. An overflow of just one LSB can cause the MSB to change states—a "white" pixel will turn "black" or vise-versa. Normally sufficient headroom is factored into the input signal and intervening computations are scaled so overflow (or underflow) does not occur. However, it may be desirable to add gain and/or offset to a low level signal to monitor the effect. It would be very annoying, in this case if the resulting "saturation" of intermittent high level pixels caused an inverse pixel response due to overflow. Such errors are aggravated by edge detect filters in an IR image detector.

Figure [18], Figure [19] and Figure [20] show graphically what happens with and without clipping. Clipping provides a saturation effect similar to analog signals.

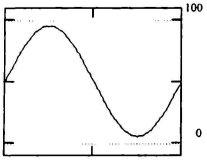


Figure [18]. Normal Signal with Headroom

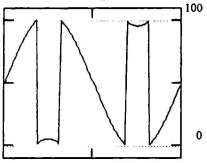


Figure [19]. Overflow Causes MSB Errors

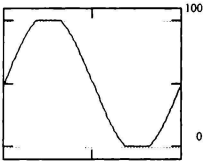


Figure [20]. Clipping Minimizes Error

Saturation arithmetic takes very little overhead. It can implemented with a few simple logic gates and multiplexer as shown in Figure [21].

Table [6] shows the clipping logic—when an overflow or underflow condition occurs, the signal is clipped to its maximum or minimum value. If no overflow/underflow occurs the signal is not effected.

INPUT			OUTPUT
A_{MSB}	B_{MSB}	C_{MSB}	
0	0	0	C
0	0	1	C_{MIN}
0	1	x	C
1	0	x	C
1	1	0	C_{MAX}
1	1	1	C

Table [6]. Clipping Logic Table

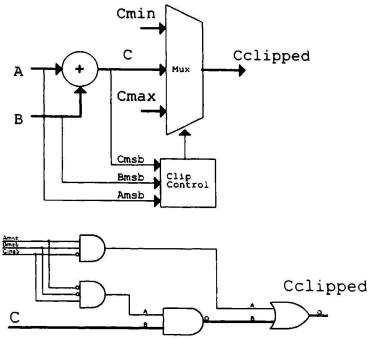


Figure [21]. Clipping Control Logic

CONCLUSIONS

The design of a high bandwidth, flexible VDP for advanced IR emitter arrays has been described. This project has included the system design, algorithms, prototype hardware design, and interface definition. The results of this effort will lead to IR scene projector drive electronics capable of controlling next generation large area emitter arrays.

ACKNOWLEDGEMENTS

This preliminary investigation/feasibility study was funded by the Naval Air Systems Command under Phase I Contract Number N68335-00-C-0121 for Small Business Innovative Research (SBIR). The results of this effort are expected to benefit joint Navy and Air Force IR Sensor Stimulation (IRSS) program at the Air Combat Environment Test and Evaluation Facility (ACETEF), Naval Air Warfare Center, Aircraft Division (NAWC-AD), Patuxent River, MD.

---

[1] Cantle, Allan J., et. al. *Ultra High Performance Image Processing Architectures for HWIL Testing*, SPIE Vol.3697, April, 1999, p.128.

[2] Joyner, Tom, et.al., *Joint Installed Systems Test Facility (JISTF) Infrared Sensor Simulation/ Stimulation Enhancement: Dynamic Virtual Reality Simulation/Stimulation Technologies For Test, Evaluation And Training*, IITSEC 1999.

[3] Olson,E.M.,et.al., *Non-Uniformity Correction of a Resistor Array IR Scene Projector*, SPIE Vol.3697, April 1999, p.405.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37282

AIAA-2000-4496

STRUCTURAL DESIGN ISSUES OF MOTION-BASED SIMULATORS

Michael J. Pagnotta, P.E.
John W. Callahan
Alex V. DiEdwardo

Pagnotta Engineering, Inc.
300 North Pottstown Pike, Suite 180
Exton, PA 19341

Abstract

Recent advances in technology have brought about great improvement in the simulation industry. The evolution of image generators, projectors, and the computers that run the training mission and control the simulator have progressed rapidly over the last ten years. The result is that motion-based flight simulators provide a realistic training environment.

While similar advances have been made in the general engineering field, mechanical requirements on a new or upgraded motion-based simulator are often based on the engineering capabilities of years past, and are sometimes derived from specifications that are 30 years old or more. The mechanical/structural portion of the flight simulator is not often viewed as a critical element in the performance of a motion-based simulator, but it is an important element in the design of a system that provides realistic training scenarios.

In order to bring the mechanical requirements of a motion-based simulator up to date with the imaging technology, two changes are necessary. First, the typical mechanical requirements on the system must be carefully reviewed to determine those that apply on modern systems and those that should be changed. Second, revised standards need to be developed based on the technology that is available.

Typical Requirements for Motion-Based Simulators

In the past decade, the principals of Pagnotta Engineering, Inc. have been involved with the design and structural analysis of many motion-based flight simulators. These simulators range from completely new systems (both the "flying platform" and the visual display system) to visual upgrades of existing wide-angle collimator (WAC) display systems. The basic mechanical requirements have always included the following:

1) A minimum resonant frequency of 7-15 Hz
2) No permanent deformation (yielding) of the structure permitted at a load level of 2-2.5 Gs in any direction with a Factor of Safety of 2-4.

The requirements may also include other parameters such as a system mass moment of inertia limit or maximum deflection. In order to determine if the requirement is a true performance parameter, one has to determine the source of the requirements.

Frequency Requirements

A system minimum resonant frequency requirement is generally set to ensure a certain level of stiffness of the simulator structure. If the frequency is specified sufficiently above the maximum input level of the actuators, load amplification from resonance of the structure at the actuator input levels would be minimized. A general rule of thumb is to specify a system minimum resonant frequency that is $\sqrt{2}$ times the peak loading maximum output frequency of the actuators. This minimum frequency keeps the load amplification to reasonable levels (a maximum load amplification of 2 occurs for a single degree-of-freedom system with no structural damping). Figure 1 contains a graph that illustrates this concept.

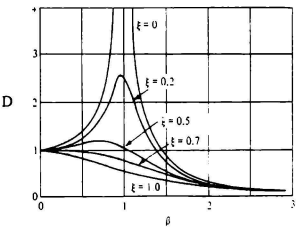


FIGURE 1

Copyright © 2000 by Pagnotta Engineering, Inc. Published By the American Institute of Aeronautics and Astronautics with permission

Although this requirement will certainly ensure that load amplification is minimized, it may be overly conservative. The basis of the requirement must first be examined. If the reason for the frequency requirement is to ensure the safety of the flight simulator's occupants, analytical tools exist to allow frequencies in the operating range of the actuators without the threat of harm to personnel. A dynamic response analysis can identify the range(s) of actuator input frequencies that will result in load amplifications. Such an analysis can also determine the magnitude of the load amplification, as long as the structural damping can be reasonably estimated. Therefore, a system can be designed to have resonant frequencies in the actuator input range without excessive load amplification.

As a specific example, refer to the previous graph. With

β = frequency ratio (applied load frequency/system free vibration frequency)
ξ = damping ratio
D = dynamic magnification factor

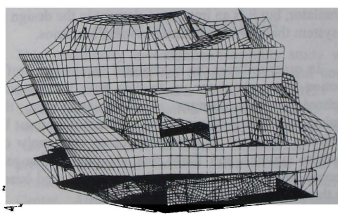
If the actuators can input 0.25 G at a frequency of 10 Hz and the system first mode frequency is 10 Hz with 10% structural damping, the maximum response of the system will be $0.25/(2*\xi) = 1.25$ G. This structural loading level may be acceptable and therefore eliminates the need for a frequency requirement.

For systems with low structural damping, where resulting load amplification may lead to an unsafe environment, another option exists. The control software for the actuators can be filtered or "notched." This process effectively bypasses the frequency ranges in which an unacceptable load amplification will occur. Current FAA Level D flight simulators are allowed to notch one band within the operating range of the actuators. Therefore, the FAA Level D simulator can have one frequency within the operating range, but the second mode must be greater than the operating frequency range of the actuators. The analysis tools in existence today can readily perform a response analysis on a complex structural system such as a motion-based flight simulator.

Personnel safety is not necessarily the only issue that drives the frequency requirement. It is apparent that the widespread growth in the use of flight simulators has occurred in part because of the advances in the visual imagery displays. Evolution of image generators, projectors, and computers has resulted in simulators that are very realistic. Thus, image quality is of primary importance in the motion-based simulator. A key component of image quality is image stability.

Image stability in terms of the simulator structure is controlled by the design of the projector support structure. The projector support structure must be sufficiently stiff so that the projectors move in unison during training. Relative projector motion (the displays of two or more adjacent projectors moving towards or away from each other) can degrade the training scenario and can cause operator sickness.

Some correlation of frequency requirement and image stability is needed to determine whether the frequency requirement adequately addresses the quality of training requirement. Referring to Figure 2 below, the deformed shape mode plot is the first resonant mode of the motion-based simulator. However, as is typical with most direct or back projection motion-based simulators, the first mode is a global roll of the structure. In this mode, the projectors may deflect from side to side, but there is little relative deformation. Therefore, image quality may not be compromised simply because the resonant frequency of the structure is within the output range of the actuators.



First mode deformed shape at 7.27 Hz
Caravan Motion Platform with Instructor Station and generic visual system.*

FIGURE 2

The frequency that will have a significant effect on image quality is the mode that results in excessive relative projector deformation and/or excessive projector deformation with respect to the viewing

* NLX Corp. Caravan motion platform with generic visual system. Printed with permission of NLX Corp.

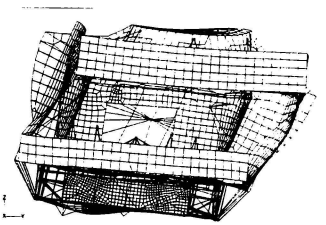
surface. Unless the local projector supports are flexible, this mode is usually much higher than the operating range of the actuators. Even if this is not so, the advances in mechanical analysis software makes it possible to determine the relative projector deformation at resonance. Such a response analysis gives deformations that can be translated into as-viewed image relative motion. As long as the relative deformation is within the acceptable range, there is no need to have a frequency requirement.

While the system frequency requirement is one way of making sure that the structure is sufficiently stiff to ensure both safety of personnel and image quality, it may be overly restrictive. Current mechanical analysis tools allow for a more quantitative evaluation of a motion-based simulator. The most important factor in revising such requirements is a thorough understanding of the specific objectives of the simulator. With the training mission in mind, the mechanical requirements can be developed making full use of current technology to obtain a safe system with the highest possible image quality. With the proper definition of the specification objectives, the simple minimum frequency requirement can be eliminated entirely.

Maximum Acceleration Requirements

The maximum acceleration requirement of 2 to 2.5 Gs is based primarily on two failure scenarios. The first is a runaway actuator mode in which, at maximum acceleration, an actuator seal will fail and the actuator extension will continue at maximum velocity until it hits its end of travel stop. The second is servo-reversal software error, in which at maximum acceleration in one direction, the actuator suddenly reverses direction, causing a rapid deceleration followed by acceleration in the opposite direction.

Both of these cases usually result in loading on the structure that far exceeds the maximum input of the actuators during normal operation. However, two issues determine if the maximum acceleration requirement is a true indication of the actual system loading. The first is the orientation of the actuators at failure. If the actuator orientation is such that the gravity vector is opposing the simulator motion at a runaway failure, the maximum loading may be lower than that specified. The type of motion that occurs during a servo-reversal must also be evaluated. A servo-reversal may be most pronounced when the simulator is relatively level, resulting in a yawing motion, or rotational acceleration about the vertical axis. This type of motion is typically not critical on simulator structures. The second issue is the location of structure relative to the motion centroid. Observing the deformed shape in Figure 3 below, if the actuator failure is such that a roll motion ensues, the lateral acceleration will vary linearly from the motion centroid outward. Therefore, acceleration of items far from the motion centroid, such as projectors, will have a higher acceleration than the overall system at its center of gravity. In such cases, the local acceleration may exceed the maximum acceleration prescribed in the specification. Again, the failure scenario helps to determine whether the specification maximum acceleration is in excess or lower than what can actually be achieved.



Deformed shape plot for lateral loading
Caravan Motion Platform with Instructor Station and generic visual system.

FIGURE 3

Factors of Safety

The factor of safety is a number greater than or equal to one and is applied to the maximum loading that can be expected in normal operating conditions (also referred to as limit loading). It is not known where the factor of safety of 2 to 4 used for most motion-based flight simulators was originally derived. A factor of safety of 4 is generally used on ground handling equipment of spacecraft and aircraft, but these types of structure usually are not weight critical and can be designed conservatively (heavy). Given that most commercial and government spacecraft use a factor of safety of 2 to yield for items that will not be tested, a factor of safety higher than 2 may be excessive and can result in an unnecessarily heavy simulator support structure. Static load testing of a simulator structure would also permit the use of a factor of safety that is lower than two when compared against permanent deformation, or yielding of the structure.

It appears to be unique to the simulation industry that the factor of safety is usually specified to be applied to a failure condition. In structural analysis, this is referred to as a fail-safe analysis. A fail-safe analysis assumes that one component has failed. That component is removed from the structural system, and the remaining structure is then assessed for the maximum loading. However, since a structural member was removed from consideration in the system, the remaining structure is analyzed with a factor of safety of 1.0, and yielding is permitted. Only catastrophic failure of the remaining structure is to be avoided.

It may be that the above criteria, while acceptable in the aircraft and spacecraft industries, is not acceptable in the simulation industry. While a fail-safe analysis will ensure the safety of personnel in or near the flight simulator, allowing permanent deformation would mean that repairs to the structure could be required if a runaway actuator or servo-reversal failure occurred. It can also be debated that a runaway actuator or servo-reversal does not constitute a true "failure" such as the loss of one bolt in a four bolt pattern or the buckling of a diagonal on a frame structure.

Consider using a factor of safety of 1.25 on the maximum loading predicted or measured due to a runaway actuator or a servo-reversal. If no yielding of the structure is permitted, and if the maximum loading envelops the expected or measured load at all locations, safety of personnel is ensured. In addition, the weight penalty associated with applying a high factor of safety for a "failure" case can be greatly reduced. The standard factor of safety required in the system specification should always apply to the normal operating conditions.

Motion-Based Simulator Specification Standards

Since many mechanical specifications have the same or similar analysis criteria such as minimum resonant frequency, maximum static loading, and factors of safety, a standard for the mechanical design of the motion-based simulator does exist. In question is the relevance of the standards that appear to the performance and safety objectives desired of the end user. Some revision or tailoring of mechanical requirements to better meet the objectives is needed.

The tailoring of mechanical standards should allow the use of the analytical tools available to better characterize the structural system. Of course, these advanced tools may increase analytical cost. However, proper up-front structural analysis prevents problems with the structure further into the program, resulting in significant cost savings.

Minimum Resonant Frequency (Stiffness)

If the performance objective is safety, the system need not have a specified minimum resonant frequency. Rather, the system could have a minimum frequency goal. If time or geometry does not permit a practical solution to meet the frequency goal, the structure should then be evaluated using a dynamic response analysis. This analysis will determine the load amplification at resonance. The structure can then be analyzed and designed for this loading. Alternatively, a limit on the response analysis could be set at no more than one-half of the runaway actuator or servo-reversal failure load. This would maintain a factor of safety of two for operating loading conditions.

Another way to avoid a specific frequency requirement is to allow a single notch, as is done with FAA Level D simulators. Again, response analysis is required to help determine the band or range of frequencies to notch. Also, all other system structural modes must be well above the input range of the actuators.

If the performance objective is image quality, the requirement can be prescribed by specifying that the resonant frequency of the structure local to the projectors (usually a platform or deck that supports all of the projectors) be at least $\sqrt{2}$ above the maximum input level of the actuators. Again, if this is too restrictive, the maximum projector deflection relative to an adjacent channel can be set to limit the on-screen image shift to an acceptable level.

Maximum Loading and Factors of Safety

Given the usually large differences between the normal operating and failure loading, it may be more practical and weight efficient to distinguish between these two conditions in the mechanical specification. A factor of safety of 2 against yielding is recommended and will help to minimize the weight penalty associated with the higher factors of safety. As long as the structural analysis accounts for local effects at connection joints, a factor of safety of 2 on the operating loads will ensure the safety of all personnel on or near the simulator.

For actuator runaway or servo-reversal failure, a factor of safety of 1.25 is recommended. This factor of safety will ensure that in the event of a failure, repairs to the structure will not be required. It will be important to quantify the failure loading. As mentioned before, structure far from the motion centroid can experience significantly higher loading than at the center of gravity of the system. Location-based loading is a way to

address this in a specification. An example of such a specification follows:

FAILURE LOADING ON THE SYSTEM:

1) 2 Gs in any direction combined with 1 g in the gravity direction for the entire system
2) (2 * X/L) Gs in any direction with 1 G in the gravity direction, where
   L = distance from motion centroid to c.g
   X = distance from motion centroid to the item in question

   for all structure where X>L

Weight and Mass Moment of Inertia

The weight of a motion-based simulator and its mass moment of inertia with respect to the motion centroid are two design parameters that are directly affected by frequency and strength requirements. Both ultimately determine how realistic the simulator "flight" is. The maximum allowable weight and mass moment of inertia is dependent on the motion actuation system and the type of aircraft being simulated. As such, it is difficult to propose an enveloping standard, since such a wide range of actuation systems and training scenarios exist. However, it is obvious that proper understanding of the true simulator requirements and translation of these requirements into the mechanical specification will minimize the potential for weight and mass moment of inertia related problems.

Challenges of Upgrading WAC Window Systems

The recent advances in technology have resulted in much more realistic training using back projection or direct projection images. Therefore, older wide-angle collimator (WAC) display systems are becoming obsolete, at least in the visual sense. In an attempt to minimize the cost of using the newer visual technology, many WAC display systems are being modified with a visual upgrade. This upgrade involves removing the WAC displays and replacing them with a direct or back projection partial dome and a projector platform and support structure. The actuator system and motion platform ("flying platform") remain, resulting in a cost savings compared to a completely new simulator. There are some challenges associated with such a visual upgrade.

Upgrade System Center of Gravity

The motion platforms on the existing WAC display systems were designed to support the system for maximum loading and balance the load on the actuators. Figure 4 shows a typical WAC display system. The center of gravity of the system is forward on the platform. By placing two of the actuator knuckles forward and one aft, the motion centroid can be placed very close to the system center of gravity, at least in the horizontal plane. In the vertical plane, the WAC display system masses are usually reasonably close to the top surface of the motion platform, which keeps the mass moment of inertia of the system low. These aspects of a WAC display system allow for a single level motion platform structure, typically 10 to 12 inches in depth.

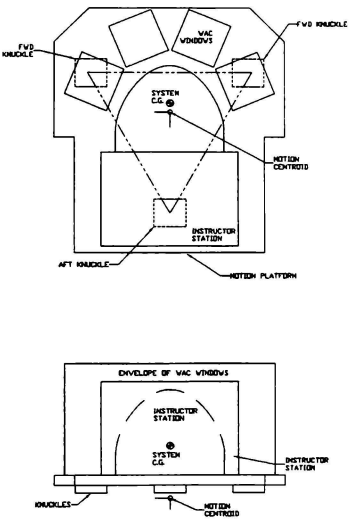


FIGURE 4

The upgrade of the WAC display system to a direct or back projection visual system poses a significant challenge to the mechanical engineer. First, much of the added mass (the projectors and their support structures)

is significantly aft of the motion centroid. Second, the added mass (the projectors, the projector support structure, and the viewing surface) raises the center of gravity of the system well above the waterline c.g. of the WAC display system. Figure 5 shows a typical visual upgrade system and the new system center of gravity.

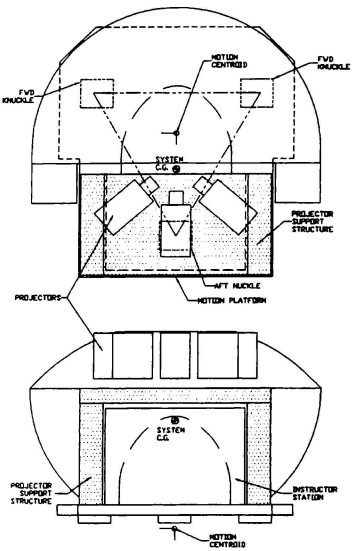


FIGURE 5

The visual upgrade results in a system that can have an unacceptable imbalance in actuator loading. A simple remedy to this problem would be to add weight forward and below of the motion centroid. However, the visual upgrade is usually heavier than the existing WAC display system, and the added weight further increases the mass moment of inertia. Too high a mass moment of inertia will affect the motion performance of the system. Therefore, the mechanical engineer may be faced with the mutually exclusive tasks of balancing the actuator loading while minimizing the system mass moment of inertia.

A logical solution to this problem is to reverse the location of the new visuals with respect to the current system. That is, the viewing surface would be supported primarily by the single knuckle and the projectors and their support structure would be supported primarily by the pair of knuckles. This configuration would allow for better balance of load on the actuators, and would also reduce the mass moment of inertia of the system with respect to the motion centroid. However, there are a few reasons why this seemingly simple fix is both complicated and costly. The first is that the control loading system would have to be relocated. Depending on the design of the motion platform, this may or may not be possible, and if possible may be costly. Another reason is that the wiring is set up for the current system, and a significant relocation of cables and harnessing may not be possible. The instructor station is also kept with the visual upgrade, and can be difficult to relocate. Finally the motion platform itself may need added members on the single knuckle end to support the wide footprint of the new viewing surface.

Another solution to reversing the system is to design a "spacer frame." The existing system could be disconnected at the actuator knuckles, lifted up and rotated 180 degrees, and would then be connected to the spacer frame. The spacer frame would connect to the actuator knuckles and provide the proper load path for the added visuals. The main drawback to this solution is that the spacer frame adds more weight and raises the system center of gravity, increasing the system mass moment of inertia.

Existing Non-Structural Operator Stations

Another challenge unique to visual upgrades is the geometry of the system and new supporting structure. As mentioned previously, instructor stations are usually left intact for use with the visual upgrade. The projectors, one of the heaviest components added, are often placed above the instructor station to optimize the on-screen viewing. Since the instructor station is typically designed to support only its own weight, it is usually not permitted to attach the new projector support structure to the existing instructor station.

Figure 5, which shows a view looking forward of a visual upgrade system, illustrates the restrictions on the projector support structure if connection to the instructor station is not an option. The resulting structure must be a moment-capable frame that wraps

around the instructor station and attaches to beams cantilevered from the motion platform. This type of design is inherently flexible in the roll direction. This flexibility is usually the dominating factor in the system minimum resonant frequency.

Using Structure Other than the "Flying Platform" for Visual Upgrade Structure Support

A possible solution to a low system frequency is to allow the projector support structure to attach to the instructor station. Other platform structures such as support frames for the cockpit and electronics (to raise the pilot eye point to the optimum position in the system) can serve as support points for the projector support structure. Any attachment permitted above the basic motion platform level will reduce the bending moment on the projector support structure side walls and increase the system frequency in the roll direction.

The main drawback to using existing structure to support the added masses, whether the projectors or viewing surface itself, is the change to the existing structure. This existing structure has been designed to support a certain weight at a particular center of gravity, and the interface to the new structure drastically alters the system primary load path. The existing structure may have been intended as a secondary structure only capable of supporting certain mass items. By including this structure in the primary load path, a re-evaluation is necessary to ensure that it has the strength required to support the added mass. If the existing structure is not capable of supporting added mass, modifications to such existing structure must be weighed against the prospect of a lower system frequency.

These tradeoffs and options must be assessed considering the earlier discussions of mechanical requirements. Again, a thorough understanding of the mechanical and performance requirements will allow the design of a simulator system that provides the most realistic training environment possible.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37283

AIAA-2000-4499

Simulating Aircraft Subsystems Using Non-Homogeneous Automata Networks

Robert Hess
Systems Engineer
BFGoodrich Aerospace
Vergennes, VT 05491

Abstract

A methodology for real-time subsystem simulation has been developed which models systems using automata networks. Such a novel modeling approach is shown to be an acceptable method for modeling aircraft subsystems suitable for use in hardware-in-the-loop operations and training environments.

Nomenclature

ΔP	differential pressure
ε	stability tolerance
F	transfer function
f	inverse function (F^{-1})
n	number of nodes
P	pressure
Q	flowrate
q	heat transfer rate
ρ	density
s	sensitivity
T	temperature
X	component internal state
U	component port variables
$\bar{}$	average of quantity
$'$	updated quantity

Introduction

Significant effort and attention has been spent on the modeling of vehicle flight controls, aerodynamics, and ground reactions. Aircrew training also requires realistic models of vehicle subsystems (fuel, hydraulics, pneumatics, gas generation systems, etc.). This is especially true in part task trainers, where the focus is not on flight training, but rather, on procedure familiarization and training. The ability to perform realistic subsystem simulation is also required to support hardware-in-the-loop (HITL) operations and testing.

Subsystem modeling is typically performed by decomposing and reverse-engineering subsystem descriptions found in vehicle flight manuals and block diagrams. Such resulting models are represented as logic tables and software state machines. As aircraft subsystems become more complex and integrated, the resulting logic / state machine models become more difficult to develop and support, especially for continuous systems.

The need existed to develop a methodology for real-time subsystem simulation that alleviated the problems associated with developing complex subsystem simulations. This methodology should:

- Be easy to reconfigure / data-driven
- Provide real-time performance
- Integrate with existing ADVISE™[1] hardware-in-the-loop environment
- Be component-oriented
- Be extensible
- Provide moderate fidelity / quasi-steady operation

Technical Approach

The initial approach for developing an environment for subsystem simulation was to use existing off-the-shelf simulation packages. These were found to be less than acceptable for a number of reasons:

Copyright © 2000 by Simmonds Precision Products, Inc. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

- Require very detailed model formulation
- Not easy to 'prune' inactive loops
- Utilize block-solver solutions
- Lack of platform support
- Not real-time capable
- Not integrable with existing software

This led BFGoodrich to look for a different approach to subsystems modeling. In order to develop this approach, three basic questions were posed:

- How do real-world components interact?
- What fundamental behavior is true of real-world components?
- Can a methodology be developed which allows system modeling using these interactions and behavior?

Component Interaction

Real-world components have inherit internal transfer functions (F) which relate the system variables of the component:

$$U_j = F\left(U_{i|i\neq j}, X_k\right) \tag{1}$$

An example of such relationships is evidenced in a simple boost pump model:

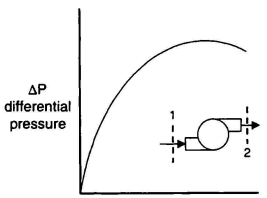


Figure 1 - Simple pump transfer function

Note that

$$Q_1 = -Q_2 = Q^{\dagger} \tag{2a}$$

† By definition, flowrate and similar properties have positive flowrate into a component, and negative flowrate out of a component.

$$\Delta P = P_2 - P_1 \tag{2b}$$

For a simple pump, ΔP is related to Q. Likewise, Q is related to ΔP. There is no preferred direction or order associated with defining component operation. Real-world pumps simultaneously vary pressure and flowrate in response to internal state changes (pump on/off/failed) and changes external to them (sudden pressure loss downstream, choked-flow, etc.). A key concept about real-world components is that they are always balanced / consistent. This balance / consistency is defined as *stability*.

A simple closed-loop fluid system can be constructed by connecting the simple pump to a simple orifice:

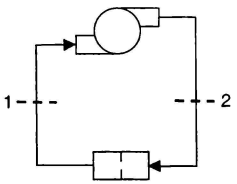


Figure 2 - Simple pump / orifice network

For such a system, the solution for pressures and flowrate through the system are given by the intersection of their respective transfer functions.

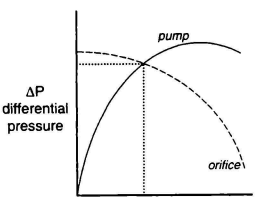


Figure 3 - Pump / orifice transfer functions

From each component's perspective, there are as set of port variables at each connection. If the system has achieved a stable solution, they are consistent, i.e.,

$$P_{1|pump} = P_{1|orifice} \quad (3a)$$

$$P_{2|pump} = P_{2|orifice} \quad (3b)$$

$$Q_{1|pump} = -Q_{1|orifice} \quad (3c)$$

$$Q_{2|pump} = -Q_{2|orifice} \quad (3d)$$

Assuming that the initial port variables are not at a stable solution, a procedure can be developed to update the port variables for each component (k):

$$\overline{Q}_k = \frac{Q_{1|k} - Q_{2|k}}{2} \quad (4a)$$

$$\overline{P}_k = \frac{P_{1|k} + P_{2|k}}{2} \quad (4b)$$

$$P'_{2|k} = \overline{P}_k + \frac{F(\overline{Q}_k)}{2} \quad (4c)$$

$$P'_{1|k} = \overline{P}_k - \frac{F(\overline{Q}_k)}{2} \quad (4d)$$

After the updates are computed, each component is stable. However, the other requirement for system solution, interface consistency, may not be achieved. A procedure can then be performed at each interface (j):

$$P'_j = \frac{1}{n}\sum_n P_j \quad (5a)$$

$$Q'_k = Q_k - \frac{1}{n}\sum_n Q_j \quad (5b)$$

The values computed at the interfaces are then used to update the port variables for each component.

A stability test can be performed for each component:

If ($|F(Q) - \Delta P| < \varepsilon$)
  Component is *Stable*
Else
  Component is *Unstable* (6)

If all components are stable, the system is allowed to move forward in time. If the system is unstable, equations (4) and (5) are repeated for each component and interface. At this point the stability test (6) is repeated. This process is repeated until the system is stable, or until some predefined number of attempts have been completed. The state transition diagram for this processing is presented in Figure 4.

This iterative technique is similar to the relaxation method[2] used to solve boundary value problems in continuum mechanics. The major difference between this approach and the relaxation method is that vehicle subsystem problem is inherently non-homogeneous and not characterized as finite-difference problems.

The processing can be grouped physically. Equations (4) and (6) are performed at the component-level, while equation (5) is performed at the interface level. By definition, the components which compute equations (4) and (6) are defined as *devices*, while the interfaces processing of equation (5) is performed by components called *nodes*. Note that devices can perform their processing based only on the information at their interfaces (port variables), while nodes perform their processing using only the immediate port variables from their connected devices. Processing can be reduced to a series of local operations. This approach does not require a specific formulation of the system into a set of equations, nor does the solution of the system require a block-solver algorithm. Because the method isolates specific processing to specific elements within the system topology, we can recast

the problem in the form of an automata network.

Automata

Automata are discrete or continuous-valued state machines connected to one another spatially. Topologically, each automaton is connected to its nearest neighbors and shares information only with its nearest neighbors. Processing by each automaton may proceed sequentially or in parallel.

The most commonly used automata are cellular automata (CA)[3]. CAs represent the simplest class of automata networks, composed of discrete-valued automata connected in simple Cartesian grids. In CA networks, all CAs are identical (homogeneous). An example of a CA used in the aerospace industry is the lattice-gas CA is used to solve hydrodynamic problems[4]. CAs have also been used to solve continuous-valued problems associated with simulating systems governed by partial differential equations[5].

Automata networks may be used to describe non-homogeneous hybrid systems[6][7] (i.e., mixed discrete / continuous-valued systems). Such modeling has been used to support formal method-based requirements verification[8][9]. The ability of automata networks to model physical systems in real-time was not the focus of such efforts.

Implementation

The resulting simulation environment developed by BFGoodrich is known as the Cooperating Object Simulation (COS). The COS has been implemented in C++. This was done to utilize the object / component-oriented structure of the automata network. As with other applications within the BFGoodrich ADVISE™ software suite, the COS is fully data-driven. The transfer functions for each device are characterized as function tables (e.g., ΔP vs. Q, ρ vs. T) or as coefficients (e.g., c_v, c_p). The stability tolerance parameters (ε) are also specified as part of the component data tables.

Libraries of specific device and node types have been implemented for the COS. The device models include pumps, valves, pipes, orifices, tanks, simple engine, heat exchangers, compressors, and fans. The node models include fluid connections (multi-phase flow), electrical connections, discrete control, analog control, rotational-mechanical and translational-mechanical motion. The COS has been design to allow new components to be added with minimum coding effort.

Besides conventional subsystem plant models, components that represent control systems, aerodynamic models, equations of motion, and related flight simulation models have been implemented. Traditional flight simulation models are easily implemented in the COS. The transfer functions in such components are simply the traditional series expansions used for aerodynamics or the differential equations of motion. By definition, such components are always stable. They perform their device transfer function computation, followed by node processing (analog nodes are typically used). As the components are stable, the integration by the force and moment components is allowed to proceed in a manner equivalent to traditional flight simulation methods.

Simulation Stability and Accuracy

Time Synchronous Operation – The COS has been designed to support multi-process / multi-processor operation. Exchange of port variable information is through the use of a shared memory area. Though the ADVISE™ software environment provides a common clock utility to support task time coordination, the iterative aspects of the COS imply a level of asynchronous operation between various COS processes. It has been shown that automata networks can achieve differing solutions if they are operated asynchronously[10]. Testing of the COS indicates that this is not a problem for the COS.

Stability and Limit Cycle Oscillation – Some of the first complex subsystems simulated with the COS displayed limit-cycle oscillations. Pressure and flowrate at specific nodes would oscillate about their predicted value. Further investigation showed that such oscillations occurred when

components with differing sensitivities (s, e.g., $\delta P/\delta Q$) were connected together via a node. Small changes in node-updated flowrate would cause large changes in pressure differential across one device. As pressure was updated at the node, the new pressure would influence a large change in flowrate at the other device. The node would then drive the flowrate in the other direction in response to the large flowrate change demanded by the second device, initiating a limit cycle oscillation. Once this phenomena was identified, a modified node-update equation was developed using a weighted flowrate update scheme:

$$Q_k^{\cdot} = Q_k - \frac{1}{n}\sum_n Q_j \frac{\left(\sum_n s_j - s_k\right)}{\sum_n s_j} \qquad (7)$$

This weighted update scheme removed the limit cycle oscillations.

Propagation Speed – As with relaxation methods, the evolution of a solution is related to the speed at which various components approach stable solutions. Speed is effected by two factors:

- Size of Network – Signal propagation through the any leg in the network is of the order

  $$1/\sqrt{2}^n \qquad (8)$$

  where n is the number of nodes in the leg of the network.

- Sensitivity of Components – Convergence is inherently slow for certain classes of multi-port components (such as wyes) because the behavior of one port is relatively insensitive to variations experienced across other ports.

Conservation – At any stable step, the error in the transfer function F is $O(\varepsilon)$. If F represents a state derivative (i.e., flowrate, rate of heat transfer, etc.), the error in propagating mass or energy at any step is $O(\varepsilon\,\Delta t)$. Over time, the mass or energy of a closed system will diverge – mass and energy conservation is not guaranteed. For many cases, conservation is not required (normal flight). However, there are classes of problems where conservation is important (refuel / defuel operations, thermal management analyses, etc.). Two approaches were considered for improving conservation:

- Decreasing Stability Tolerance ε – Decreasing ε would reduce the propagation error at each step. However, this would also increase the time required to achieve a solution.

- Provide a Global Conservation Mechanism – A mechanism was implemented to check conservable properties at the start and end of each frame. If there was a change, the error was redistributed across those components that contributed to the error. This required each device to determine if it was *active* during the time frame (activity can be defined simply as being on, not failed, etc.). To make the conservation operation more efficient, devices also have the ability to determine if they have conservable quantities. A flight control system would not have any conservables, whereas a fuel tank would.

Sample COS Operation

Figure 5 illustrates portions of an advanced integrated fuel / ECS system which was modeled in ADVISE™. In this system, the integration of five distinct subsystems has been analyzed to insure proper component sizing and behavior.

The basic heat sink for the system is fuel. For the purposes of this discussion, the fuel is denoted as fluid 1. Fuel is delivered from the feed tanks (TANK1 and TANK2) to the engines (ENGINE1 and ENGINE2) through a modulating valve (V1). Prior to reaching the engines, the fuel absorbs heat from three liquid-liquid heat exchangers (HX3, HX4, and HX5). HX3 allows fluid 3 to transfer heat to the fuel, while HX4 and HX5 allows fluid 2 to exchange heat with the fuel (see Table 1 for the thermodynamic properties of all the fluids). Note that

TANK1, P1, HX4 and ENGINE1 are identical with TANK2, P2, HX5, and ENGINE2. Also note that the check valves in both sides of the fuel system are identical.

There are 3 primary sources of thermal load outside of the fuel system. They are the avionics compartments (Q1, Q2, Q3), fan heating (F), and pump heating (P3, P4). Three of these loads add heat to fluid 5, two of these loads add heat to fluid 4, while the remaining loads add heat to fluid 3. (See Table 2 for the heat loads associated with Q1, Q2, and Q3). Examination of Figure 5 reveals that there are 3 flow networks external to the fuel system, two of which exchange thermal energy to fluid 3.

This system was simulated with the COS. Fluid networks 1 and 2 were combined together in one simulations, fluids 3 and 4 were combined into a second simulation, and fluid 5 was partitioned in a third simulation. All three simulations ran simultaneously within a single ADVISE™ session (the first simulation operated at 2 Hz, the second at 5 Hz, and the third at 10 Hz). This simulation was initialized with fuel tank temperatures and compartment temperatures set at 520 degrees R (60 degrees F). The simulation was allowed to run until the system came to a steady-state condition.

Comparison to Theoretical Response

The COS response for this analysis is presented in the following paragraphs. Note that the critical flowrates (Q), temperatures, and pressures are provided. Also note that the heat transfer for each heat exchanger predicted by the COS is also provided.

Fluid 1 Network

Engine feed flow is provided from the feed tanks. The feed tanks provide flow at 520 degrees R. The tanks are vented to the ambient (14.7 psi), and hydrostatic effects are neglected. Hence, the fluid pressure at the tank outlets should match the ambient pressure.

Identical boost pumps (P1 and P2) boost the fuel flow to 38.8 psi. For these pumps, the theoretical pressure rise is given by:

$$\Delta P_{theory} = 30 - Q_P \text{ psi}$$

For the predicted fuel flow through each pump (5.85 pps), the theoretical pressure rise is

$$\Delta P_{theory} = 30 - 5.85 \text{ psi}$$

$$= 24.15 \text{ psi}$$

which agrees very well with the pressure rise predicted by the COS (38.8 - 14.7 = 24.1 psi). This same analysis was completed for all the components in the fuel network, and the appropriate pressure / flow relationships were observed.

In this system, fuel is being used as thermal mass. The fuel is heated due to pump heating (5.7 deg. R rise), the transfer of heat from fluid 2 (heat exchangers HX4 and HX5), and the transfer of heat from fluid 3 (HX3). The theoretical temperature rise from the feed tank to the engine inlet is calculated by:

$$\Delta T_{theory} = \Delta T_P + \Delta T_{HX3} + \Delta T_{HX4}$$

$$= \Delta T_P + q_{HX3} / (Q_{HX3}\, c_p) + q_{HX4} / (Q_{HX4}\, c_p)$$

$$= 5.7 \text{ deg. R} + 5.99 / (11.7) / (0.46) \text{ deg. R} + 23.07 / (5.85) / (0.46) \text{ deg R}$$

$$= 15.4 \text{ deg. R}$$

Which is in agreement with the temperature rise predicted by the COS (535.4 - 520 = 15.4 deg. R).

Fluid 2 Network

Heat exchangers HX4 and HX5 transfer heat from fluid 2 to the fuel (fluid 1). The theoretical hot-side efficiencies for these heat exchangers are 10%. Given the hot-side and cold-side inlet flows, the hot-side outlet flow is computed by:

$$T_{theory} = T_{hot,in} - \eta\,(T_{hot,in} - T_{cold,in})$$

$$= 700 \text{ deg R.} - 0.1\,(700 - 540.4) \text{ deg. R}$$

$$= 684 \text{ deg. R}$$

The COS computed value was 682.7 deg. R, which is within 0.2% of the theoretical value. Analysis of the heat transfer from the hot-side to the cold-side also confirms the heat rise on the cold-side outlet.

Fluid 3 Network

In this network, heat from the avionics compartments are transferred to fluid 3 from HX2 (heat exchanger 2) and HX1 (heat exchanger 1). In addition, fluid 3 is heated by pump 4 (P4). Heat leaves the network at heat exchanger 3 (HX3)

Based on the pump efficiency, the pump heating is anticipated to be approx. 0.7 degree R.

For this network the thermal balance is characterized by:

$$q_{HX3,theory} = q_{HX1} + q_{HX2} + q_{P4}$$

$$= q_{HX1} + q_{HX2} + Q_{P4}\, c_p\, \Delta T$$

$$= 1.15 \text{ BTU/sec} + 2.30 \text{ BTU/sec} + (6.26)(0.575)(0.7) \text{ BTU/ sec}$$

$$= 5.97 \text{ BTU/sec}$$

This agrees with the COS prediction of 5.99 BTU/sec within 0.4%. Also note that the COS maintains a flow balance through the network (no flow is lost or created through the branches of the network).

Fluid 4 Network

In this network, heat from a single avionics compartment is transferred to fluid 3 at HX2 (heat exchanger 2). In addition, fluid 4 is heated by pump 3 (P3). Each component in this network (excluding the pump) was selected to provide a 1 psi pressure drop at 1 pps of flow. Theoretically, the pump should provide a 3 psi pressure rise at 1 pps of flow.

Based on the pump efficiency, this heating is anticipated to be approx. 0.5 degree R.

For this network the thermal balance is characterized by:

$$q_{HX2,theory} = q_{Q3} + q_{P3}$$

$$= q_{Q3} + Q_{P3}\, c_p\, \Delta T$$

$$= 2 \text{ BTU/sec} + (1)(0.575)(0.5) \text{ BTU/sec}$$

$$= 2.29 \text{ BTU/sec}$$

This agrees with the COS predication of 2.30 BTU/sec within 0.5%. The COS also matches the theoretical pressure/flow relationship for this network (1 pps flowrate with a 3 psi pressure drop).

Fluid 5 Network

In this network, heat from two avionics compartments is transferred to fluid 3 at HX1 (heat exchanger 1). In addition, the fan heats fluid 5. Based on the fan efficiency, this heating is anticipated to be approx. 1 degree R.

For this network the thermal balance is characterized by:

$$q_{HX1,theory} = q_{Q1} + q_{Q2} + q_F$$

$$= q_{Q1} + q_{Q2} + Q_F\, c_p\, \Delta T$$

$$= 0.5 \text{ BTU/sec} + 0.5 \text{ BTU/sec} + (0.63)(0.24)(1) \text{ BTU/sec}$$

$$= 1.15 \text{ BTU/sec}$$

This number is consistent with the value predicted by the COS. In addition, the theoretical pressure/flow relationships match those predicted by the COS.

Conclusions

An environment for modeling complex systems using automata networks was developed. The environment, known as the ADVISE™ Cooperating Object Simulation (COS), has been shown to be useful for subsystem simulation. The COS is an object-oriented simulation that represents subsystems as hybrid automata networks.

The nature of the COS allows it to support real-time operations, while at the same time provide adequate fidelity for many real-time applications. Comparisons of theoretical and COS-predicted response show that COS has excellent quasi-steady prediction capabilities.

There are a number of benefits in using the COS:

- Decomposing a system into a collection of related automata provides an intuitive, object-oriented environment that allows the user to create a system as a collection of components, not as a collection of equations.

- The crisp component boundaries of automata allow complex, integrated system models to be partitioned into multiple simulations, thereby allowing the system to be simulated using multiple processes / processors. Such an approach also allows the user to limit the effects of poorly defined models and systems on the total system operation.

The main disadvantage of the COS is associated with information propagation speed through the automata network. This is becoming less of an issue as computer speeds increase.

Bibliography

[1] Hess, Robert, "A Component-Oriented Environment for Systems Development", 14th AIAA/IEEE Digital Avionics Systems Conference, Nov. 1995.

[2] Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, 1992.

[3] Weisbach, Gerard, Complex Systems Dynamics, Addison-Wesley Publishing Company, 1989.

[4] Boghosian, Bruce, "Lattice Gases and Cellular Automata", Elsevier Press, 1999.

[5] Ostrov, Daniel, "Continuous Valued Cellular Automata for Nonlinear Wave Equations", Complex Systems 10, 1997.

[6] Henzinger, Thomas, "Theory of Hybrid Automata", 11th Annual IEEE Symposium on Logic in Computer science, pp. 278-292.

[7] Traktenbrot, Boris, "Automata and Hybrid Systems", Lecture Notes presented at Uppsala University, Autumn 1997.

[8] Archer, Myla, Heitmeyer, Constance, "Verifying Hybrid Systems Modeled as Timed Automata: A Case Study", presented at HAART '97, March 1997.

[9] Heitmeyer, Constance, Lynch, Nancy, "The Generalized Railroad Crossing: A Case Study in Formal Verification of Real-Time Systems, proceedings of IEEE Real-Time Systems Symposium, Dec. 1994.

[10] Huberman, Bernardo, Glance, Natalie, "Evolutionary Games and Computer Simulations", Proceeding of Natl. Academy of Science.

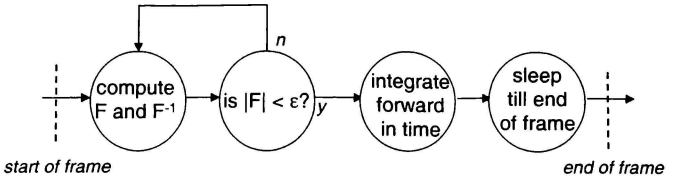


Figure 4 - COS processing methodology

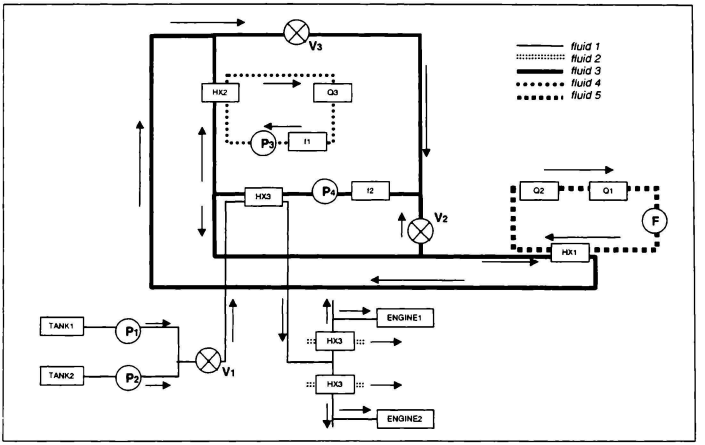


Figure 5 - Coupled fuel - environmental control system (ECS) network

Table 1 - Fluid Thermodynamic Properties

FLUID	DENSITY (lbm/ft^3)	SPECIFIC HEAT (BTU/lbm/degR)
fluid 1	57.0	0.46
fluid 2	50.0	0.40
fluid 3	66.0	0.575
fluid 4	66.0	0.575
fluid 5	0.76	0.24

Table 2 - Heat Loads and Steady-State Compartment Temperatures

COMPARTMENT	HEAT LOAD (BTU/sec)	STEADY-STATE TEMPERATURE (degR)
Q1	0.5	616.7
Q2	0.5	610.0
Q3	2.0	580.0

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37284

AIAA 2000-4500

DATA MANAGEMENT IN THE ASSET SIMULATION FRAMEWORK

Jeffrey Maddalon*, Stephen Derry*
{j.m.maddalon | s.d.derry}@larc.nasa.gov

NASA Langley Research Center
Hampton VA, 23681

Abstract

The Aircraft System Simulation Environment and Toolkit (ASSET) is a software framework for rapid development and simulation of continuous systems. The central concept of ASSET is a model—analogous to a block in a system block diagram. One important way to simplify the software development process for models is to use *encapsulation*. Encapsulation is a software development technique that disallows one model from accessing another model's internal state; models can only communicate through their well-defined input/output interface. Unfortunately when building a simulation, the engineer will often require visibility to data values that are not part of the normal interface. The data management system of the ASSET framework attempts to solve this problem. The data management system consists of a large central repository in which all models may write data and a common set of tools to access the data.

Introduction

The Aircraft System Simulation Environment and Toolkit (ASSET) is a software framework for rapid development and simulation of continuous systems. The framework was designed primarily to simulate aircraft and aircraft systems. ASSET is entirely written in the Java computer language from Sun Microsystems [1]. The Java computer language offers advantages over other languages in the areas of portability, support for the object-oriented design paradigm, and ease of learning.

The central concept of ASSET is a model [2]—analogous to a block in a system block diagram. Models have well-defined input and output vectors and may contain states and/or other models. A system can be simulated in ASSET with one or more models. Other parts of ASSET include a central data repository, software to present simulation data in a useful format and a graphical user interface to ease user interaction. In an environment with rapidly changing requirements, models often need to change frequently. To reduce the impact of this, one important goal of ASSET is to minimize the work of the model designer.

One method to achieve the goal of decreasing the work of the model designer is to protect a model from other models. The model designer must define the input and output vectors for the model, and these vectors are the *only* interface between the model and the world outside. Model designers do not need to concern themselves about how or when other models will read their data since models within ASSET can only access the well-defined interface of the model. The technique of isolating models from each other is termed *encapsulation* [3].

Software frameworks that do not implement encapsulation usually encounter difficulty maintaining scalability. As the system grows and more models are added, it becomes much more likely that one component will supply data to another component incorrectly. Models with a well-defined interface (that is, models with well-defined input and output vectors) should also be more reusable than a model without encapsulation. If someone wants to use an encapsulated model in another context, then all they need to provide are the input signals. Since models are guaranteed to be independent of each other, they should be able to be moved to many environments (such as batch or real-time). Finally, a well-defined interface allows model writers the freedom to change the implementation of a model without forcing the users of this model to change their software.

* *Computer Engineer, Systems Development Branch*
Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the copyright owner.

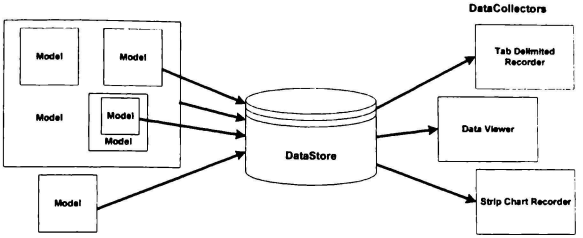


Figure 1 - ASSET Data Management System

To effectively implement encapsulation, the model designer should attempt to minimize the number of both inputs and outputs. Building a model with too many input signals brings all the problems of lack of encapsulation within the model and unnecessarily increases complexity on the model's owner. A model with too many outputs unnecessarily exposes the model's implementation and reduces its reusability.

This encapsulation technique provides powerful leverage for the ultimate scalability and reusability of the ASSET framework; however it comes at a cost. Often when developing a system, the designer wants to know not only the values of the input and output vectors but also the value of intermediate computations and states for any model in the system. These values may not be available through the output vector. The designer could add "test points" to the output vector; however, to maintain backward compatibility, every future implementation of this model must produce these signals—even if they are not relevant to the current implementation. In addition every time the designer wants another signal he or she must change the code and add this signal and recompile. Finally, the designer must actually have the model's source code and even if it were available, the designer would need a detailed understanding of the implementation to know where to place the test point. Is there a better way?

In the above scenario, one may recognize two distinct roles: the system designer who is concerned with simulating the whole system and the model designer who is concerned with one particular subsystem. Oftentimes, the same person fills both of these roles but as the project becomes larger, it is less likely that one person will be able to fill them. As mentioned, system developers are concerned about how the whole system operates. They may be concerned with a particular subsystem, until they are convinced that it is operating properly, then they will turn their attention to another subsystem. Model developers are in the best position to determine what signals are truly required to be the inputs and outputs of the model. Model designers can also make the best decision about what internal values may be of interest to future users given the current implementation of the model. To fully implement encapsulation, the model developer must also ensure that other models do not rely on values that are not part of the model's normal interface.

The data management system of ASSET recognizes the interplay between the two roles: the system developer who needs signals from all over the system and the model developer who needs models with the smallest interface possible. ASSET's data management system provides a structure such that each designer can effectively perform his or her job.

Overview

ASSET's data management system separates the data generators from the data consumers. Data generators are usually models and data consumers are called DataCollectors. The basic design is that models write data into a central repository, called the DataStore, which stores a time-history of the data. Once the data is in this repository, the DataCollectors can then pull data out. Figure 1 illustrates the relationship between Models, DataStore, and DataCollectors. This data flow is strictly in one direction. The data management system does not provide any facility to retrieve data from either DataCollectors or the DataStore and insert this data into a model.

The design of the data management system supports the model developer and system developer roles, previously described. Since model designers are in the best position to determine what values are of interest in the model, they decide what data the model can send to the DataStore. System designers decide if that model should send any data to the DataStore. In other words, those who use

the model (the system designers) decide if the data from that model is of any interest.

The data management system also enjoys the benefits of encapsulation without incurring its penalties. Since there is no ability to feed data back into the models, model designers are assured that signals cannot be incorrectly injected into their models. Signals must come in to models through their input vector. In addition, model designers do not need to unnecessarily expose their implementation to allow system designers to examine the values they find interesting. The model designers also do not need to be concerned with who might want the data from their models and where to send that data. They send the data from their model to the central collection point, where the system designer can view it. The data management system provides a small "back door" through this strict encapsulation interface to allow system designers to gather the information necessary to do their job.

In addition to supporting the needs of the two types of ASSET designers, a few other interesting features come out of its design. Since all data in an ASSET simulation is stored in the same format in the DataStore, creating a structure that allows data presentation in any format is almost trivial. Through the object-oriented concept of inheritance, all DataCollectors share the same software that manages a group of channels and accesses the DataStore. If a user requires a new data format they can derive from the general DataCollector hierarchy. In this way, they only need to be concerned with the specific requirements of the data format, not the details of pulling data out of the DataStore. The concept of a data format is quite broad; in the small sense, it could mean a tab-delimited text format or a binary format, but it could also mean a graphical strip chart system or even a graphical flight display.

Also, the creation of a central repository allows the DataCollectors to be generic. The DataCollectors do not have to be aware of the interface of every model whose data they use. They only need to know what data they want, not which model generated it. The separation between models and DataCollectors goes even further. Models can run asynchronously from the DataCollectors. Synchronizing the data is accomplished in the common DataCollector software. This allows the data recording to proceed when the main computation is idle or if an ASSET simulation was running on a multiprocessor computer, on another processor.

The three major components of ASSET's data management system are the models, which generate the data, the DataStore, which serves as the central repository, and the DataCollectors, which display the data in some useful format. Each of these components is described in its own section below.

Models and Their Data

One important concept is the balance between the model designers who are interested in being insulated from the context in which their model operates and the system designers who are interested in getting all the data needed to do their job.

Since the modeler cannot know what signals the system designer will be interested in, the design of ASSET allows the modeler to specify all signals that may be of interest. These signals may include inputs, outputs, parameters, states (continuous and/or discrete) or intermediate computations. Once the modelers complete this specification they no longer need to be concerned with data management. The Model superclass [2] and the DataStore perform all necessary operations to actually get the data out of the model. Specifically, the Model superclass determines when the signals in the model are ready to be sampled. The DataStore maintains a time-history of each signal.

System designers are also aided by the design of the data management system for ASSET. These developers only need to indicate that they are interested in values from this model. If they are not interested in values from this model, then no computational penalty is incurred. If they are interested in values from this model, then they automatically get access to all values that the modeler specified. System designers are not required to view these signals if they do not want to, but they are available. System designers also have the ability to set the sampling rate. This is an integer representing a sub-division of the simulation frame rate. If the sampling rate is 3, then the data will be sampled every third frame. The DataStore has no knowledge of how often models write data into the DataStore. Since the DataStore has no concept of this sampling rate, the sampling rate can be changed while the simulation operates, including being set to zero which turns sampling off.

A class indicates that it is able to write data into the DataStore by implementing the Recordable interface†. The Model superclass implements this interface; therefore all ASSET models can record data in the DataStore. The Recordable interface contains two methods. The `setu-`

† An interface is a special Java construct, similar to an abstract class, which only contains method interfaces but not their implementations [1].

```
public void setupRecordable(int objectID) {
   datastore.addChannel(objectID, "speed", "ft/s");
   datastore.addChannel(objectID, "alt", "ft");
   datastore.addChannel(objectID, "alpha", "deg");
   datastore.addChannel(objectID, "beta", "deg");

   super.setupRecordable(objectID);
   }

public void record(int channelID, double[] values) {
   values[channelID++] = speed;
   values[channelID++] = altitude;
   values[channelID++] = alpha;
   values[channelID++] = beta;

   super.record(channelID, values);
   }
```

Figure 2 – Sample Recordable Methods.

`pRecordable()` method performs setup operations for a channel such as specifying a name and providing suggested units for the DataCollectors to use. The `record()` method actually copies the data into the DataStore. One method could be used for both functions; however, that would incur a greater performance cost. With the current structure, only the `record()` method needs to be called during the main execution loop of the simulation. An example of the two methods of the Recordable interface is given in figure 2.

The Model superclass tells the DataStore when the signals in the model are ready to be recorded. This operation involves calling the DataStore's `sample()` method. The Model gives the value of its simulation time to DataStore with this method. The DataStore will then invoke the model's `record()` method. Through this method, the DataStore will pass the Recordable an array. The Recordable must then fill this array with the appropriate values going to the DataStore. When the `record()` method completes, the DataStore can alert any collectors that new data has arrived.

ASSET models are the major implementers of the Recordable interface; however, any class that implements the Recordable interface can add data to the DataStore. An example of a class that may be a Recordable but is not a model would be the interface software for a cockpit simulator. The system designer may want to know exactly what is coming out of the hardware, before any modeling software uses the data. Currently in ASSET, Models are the only classes that implement the Recordable interface; so for the purposes of this paper, the terms model and Recordable are interchangeable.

DataStore

The DataStore is built as a series of channels, where each channel is a structure that contains a time history of a particular data value. At a particular instant in time, models will generate the data value and notify the DataStore. The DataStore will retain the time history of this quantity, thus creating a channel. By maintaining a history of each data value, the DataStore effectively decouples the data collection from the data output—allowing models to execute at a different rate from the DataCollectors.

The DataStore serves as the central repository for data in an ASSET simulation. Models write their data into the DataStore, and DataCollectors read data out of the DataStore and present this data to the user. This design has several features. The models do not need to make their interface available to every model in the simulation—they only need to make their interface available to those models that truly interact with the model. The data entry portion of DataStore is performance sensitive, but the DataCollectors are not; therefore, the data entry software can be tuned for a real-time environment. This partitioning allows the main simulation to operate independently of the data collection.

At its core, the DataStore is simply a three-dimensional block of `doubles`, with the dimensions of models, quantities, and time. (All quantities in ASSET are stored as a `double`, a data type in Java indicating a double precision floating point value.) Since the purpose of the DataStore is to store channels (and a channel is the time history of a quantity), the dimensions of *quantity* and *time* fall out naturally. The dimension of *models* is

needed because some information is the same for all channels within a model. The most prominent member of this group is the value of time. The DataStore stores a separate time sequence for each model. This allows the models to operate at different rates and even asynchronously from each other. Furthermore, since a value of time is stored for each dimension, the models can change their update rate, at any time without affecting the DataStore. The DataStore does not provide any correlation of the signals from different models—this is the job of the DataCollectors.

At construction-time the user specifies the size of the DataStore. This size is the approximate number of Java `doubles` that the DataStore allocates. Nominally, the DataStore will divide this space equally among the channels. For a simulation of a given complexity, the size of the buffer indicates how many past data values the DataStore can hold; the greater the size, the longer the time history can be stored for each channel. The DataStore cannot derive a time (how many seconds of data can be held) from this parameter, because then the DataStore would know how often a model is going to write data into the DataStore. The rate at which a model writes data is entirely up to the model itself. The model can change this rate (and even stop it) at any time. Once the buffer within the DataStore is exhausted, the DataStore will write over the oldest data. The DataCollectors must ensure that they acquire the data they need before the data is overwritten. Since the DataStore operates in the main simulation loop and allocating memory[‡] during this time would probably preclude the use of ASSET in a real-time environment, the DataStore does not allocate any memory when running in the main simulation loop.

If one model's iteration rate is twice as fast as other models in the system, then this model will need twice as much storage as other models to be able to store a time-history of the same length. It is planned that the user will be able to weight the data allocation within the DataStore to account for this situation. Currently this feature is not implemented.

When the DataStore has all the data for a particular model at a particular time, it reports the arrival of this new data to the DataCollectors. The DataStore only alerts the DataCollectors; it does not send them any data. It is the DataCollectors' responsibility to determine if they are interested in the data and to read the data from the DataStore. The DataStore can alert the DataCollectors in one of two ways: as an inline method or through a Java event. Currently only the inline method is implemented. The inline technique assumes a single thread of control will flow from a model generating data, to the DataStore recording this data, and then to the DataCollectors displaying this data. The event technique provides a mechanism for the models and DataCollectors to operate independently of each other. One thread (associated with a model) can write data into the DataStore and not interfere with another thread (associated with a DataCollector) reading data out of the DataStore.

The DataStore is designed to operate as independently as possible from both models and DataCollectors. One of these components can change without forcing changes on the other components. As a result the DataStore deliberately does not understand certain concepts. For instance, the DataStore doesn't have an independent concept of time. Specifically, it doesn't assume it knows what the "next" time will be. This allows the model to change the data update rate without consequence to the DataStore. Since the DataStore cannot make any assumptions about time, the DataStore does not perform any synchronization of data between different models. The DataCollectors are responsible for correlating the data coming out of the DataStore.

DataCollectors

Once the data is in the DataStore, the data output problem becomes significantly easier; all data is available in a central location and the data can be accessed through a single interface. The real problem for the design of the DataCollectors is how to develop a structure that facilitates new data output capabilities with a minimal amount of work.

The DataCollectors are the classes that take data out of the DataStore and present this data in some format useful to the user. DataCollector is an abstract class; it only provides a structure to copy data out of the DataStore, not a full implementation. Since all the data in an ASSET simulation is stored in a common format in the DataStore, only one software module needs to be written to take data out of the DataStore; this software is the DataCollector superclass. The DataCollector superclass does not display any data itself. To display data, a class must be built that inherits from DataCollector. ASSET provides several classes to display the data including space-

[‡] Allocating memory often involves a non-deterministic execution time. Thus, its use during a real-time simulation must be avoided.

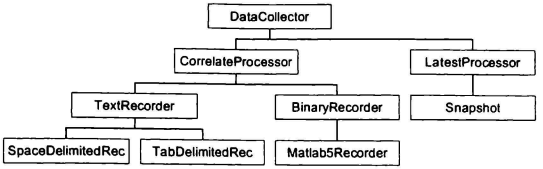


Figure 3 - DataCollector Inheritance Hierarchy

and tab- delimited recorders, MATLAB[§] recorders, and snapshots (classes that report the latest value for a channel). In the future, ASSET will provide advanced DataCollectors like graphical strip-chart plotters and graphical flight displays. The hierarchy of DataCollectors was designed to allow the user to quickly produce new software to display data in different formats.

Currently ASSET provides two classes that derive from DataCollector: CorrelateProcessor and LatestProcessor. Whereas the DataCollector superclass only understands how to take data out of the DataStore, these processor classes determine which data is processed, but they do not define how to display the data. Recognizing that models can write data at different rates, the CorrelateProcessor can correlate information between different models. The CorrelateProcessor maintains a time independent of the simulation time. It will process the data after it correlates the data from all channels for this time. The CorrelateProcessor will perform a zero-order hold on any data that has not been updated for the current time. The LatestProcessor simply creates a data set of the latest available data for all channels. Neither processor will interpolate under any circumstance. A special InterpolateProcessor is planned for such applications.

The differences between the CorrelateProcessor and the LatestProcessor can be subtle. The major difference is when a CorrelateProcessor produces a data set for time t, the data within this set is valid for time t. The LatestProcessor does not perform this correlation. If two models operate asynchronously from each other, they could have different simulation times. The LatestProcessor does not take this difference into account. Another difference between these two processors is the CorrelateProcessor can operate without intervention from the user. Once all data is available for a particular time, the CorrelateProcessor will form the data set and display the data. The user must explicitly invoke the LatestProcessor when the data is to be processed.

As discussed above, the DataStore can alert a DataCollector in one of two ways: inline and event. All DataCollectors can be alerted by either method. The inline method minimizes the amount of storage the DataStore needs to allocate and the event method minimizes the time spent in the main computational loop. With the inline method, the DataCollectors always have the data they need to display; so there is no risk that any data will be lost. Since data cannot be lost, past values do not need to be stored in the DataStore; therefore, the inline method minimizes the amount of storage. Unfortunately, this method executes data display operations during the main execution loop. These operations may involve writing to files or drawing graphics, which could have a large and nondeterministic execution time. In a real-time environment this is unacceptable. The event method minimizes the time spent in the main computational loop, but this method requires enough storage in the DataStore to buffer the data to mitigate the possibility of losing data. Using this mechanism, the when the DataStore receives new data, it simply sends a Java event to the DataCollectors and immediately returns to the main simulation loop. The DataCollectors must then access the DataStore and read out any needed data before it is overwritten. This allows the simulation to operate in real time, but the data recording, since it involves writing to the disk, can occur when the real-time computation is idle. In summary, the inline method works best for batch simulations (simulations that are not sensitive to nondeterministic execution times) and the event method works best for real-time simulations.

The table below contains examples of different processors and alerting mechanisms and examples of their use.

[§] MATLAB® is a commercial software package for computation, visualization, and programming produced by The MathWorks, Incorporated. A MATLAB recorder is an ASSET class that writes data in MATLAB's native format.

Processor	Inline Alerting	Event Alerting
Correlate-Processor	• Batch recorder • Batch plotter	• Real-time recorder • Graphical strip chart recorder
Latest-Processor	• Real-time hardware output • Batch snapshot	• Graphical data viewer • Real-time snapshot

An inheritance hierarchy for members of the DataCollector family is shown in figure 3. The leaves of this tree are classes that can be instantiated; the other members are abstract classes. A recorder is a DataCollector that writes time-history data out to a disk file. There are two types of recorders: TextRecorder and BinaryRecorder. As their names suggest, TextRecorder writes text files and BinaryRecorder writes binary files. These classes perform any necessary operations to create and manage disk files, but they do not contain any information about the format of the file. The two subclasses of TextRecorder are the space- and tab- delimited recorders. These DataCollectors write information in a simple text format with the columns of data separated by either spaces or tabs. The Matlab5Recorder will write data in a format compatible with MATLAB version 5.0 from The MathWorks, Incorporated [4]. These three classes (TabDelimitedRecorder, SpaceDelimitedRecorder, and Matlab5Recorder) essentially only contain one method to write the file in the appropriate format. If the user wanted to add a different format, they would simply inherit from either the TextRecorder or BinaryRecorder classes and define this method. A Snapshot is a DataCollector that saves the latest values of all the channels to a file. This is useful for capturing a view of the model at a particular instant of time (that is, a "snapshot").

The main configuration parameter for recorders is the recording rate. Other configuration parameters include the selection of channels, their associated units, and formatting information such as the number of decimal digits to display. The recording rate indicates how often the collector will record the data to a file. This feature allows the user to distinguish between the recording rate and the simulation rate. The simulation rate may increase or decrease, but if the recording rate does not change, the amount of data in the file will not change.

Just as the DataStore does not assume any rate that data arrives, the DataCollectors also do not assume any arrival rate for data. Generally, DataCollectors do not store any data themselves; they rely on DataStore to retain past values for each channel. The user must be careful about relying on the DataStore to store past values. When the DataStore fills all its storage for a channel, it will overwrite the oldest values. The CorrelateProcessor will detect this condition (this condition has no meaning to a LatestProcessor). To avoid this situation, the user can either use the inline method to alert the collector—incurring any resulting performance penalties—or ensure the DataStore has enough memory.

The user may add any number of channels to a DataCollector from the DataStore. There is no restriction on what data any DataCollector may receive; if the data is in the DataStore, all DataCollectors can read it. Several DataCollectors can simultaneously read the same channel without any difficulty. When the user adds a channel to the collector, one important specification is the unit to be used for displaying the quantities recorded on this channel. Since all data in the DataStore is specified in internal units [2], the quantity needs to be converted to the desired unit prior to outputting the data. If the user of a DataCollector does not specify the units for a channel; then suggested units provided by the model writers will be associated with the channel.

Status

Most of the software described in this paper for the ASSET's data management system has been in service for approximately nine months. The software to implement the event-oriented method for alerting DataCollectors has not been implemented. Using a representative simulation model (a version of the F-16 aircraft model described in [5]), the data management system (with the Matlab5Recorder) was able to record 210 signals in 35 microseconds on a 550 MHz Pentium III machine. This time just measures the performance impact of the data management system within the main computational loop of a simulation (i.e., the time to record the data into the DataStore). This time does not include much of the computational load of a DataCollector, including any time to actually write data to disk.

The software has demonstrated that new data formats can be added quickly. The authors were able to build the SnapShot collector in approximately one hour. It is expected that less experienced developers would take longer, but not excessively longer, to develop new collectors.

In addition to completing the event alerting software, future enhancements to this software will include testing this in a real-time environment. Full testing of the real-

time performance of this system will require the event alerting mechanism to be completely functional. Also the software must be verified to work within a multi-processing computer system. Achieving high performance while maintaining data integrity will be a challenge in this type of environment. The software was designed to allow multiple independent threads to write into the DataStore simultaneously; however, this operation will need to be verified.

Summary

ASSET's data management system provides a solution to a fundamental problem in simulation software development. Software can be developed faster and with higher quality if the modeler can be insulated from other software in the system. However, this model encapsulation also hinders legitimate use by others who require access to all signals so they can gain an understanding of the system as a whole. The data management system, employing a central repository, combines encapsulation with one-way visibility to meet both requirements. As a side benefit of providing a central repository for the data, new data formats can be added with very little effort.

References

[1] Arnold, Ken; Gosling, James. *The Java Programming Language, Second Edition.* Addison Wesley Publishing Company, Reading, MA, 1998.

[2] Derry, Stephen; Maddalon, Jeffrey. *Implementing Dynamic System Models in the ASSET Simulation Framework.* AIAA 2000-4393. Modeling and Simulation Technology Conference, Denver, CO, August 2000.

[3] Booch, Grady. *Object-Oriented Analysis and Design with Applications, Second Edition.* Benjamin / Cummings Publishing, Redwood City, CA, 1994.

[4] The MathWorks, Inc. *MAT-File Format, Version 5.* The MathWorks, Inc. © 1999. http://www.mathworks.com

[5] Stevens, Brian L.; Lewis, Frank L. *Aircraft Control and Simulation.* John Wiley and Sons, Inc., New York, NY, 1992.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37285

AIAA-2000-4501

SOFTWARE ARCHITECTURE FOR A RECONFIGURABLE FLIGHT SIMULATOR

C.A. Ippolito*
MITRE Corp.
McLean VA

A.R. Pritchett†
Georgia Institute of Technology
Atlanta GA

ABSTRACT

While simulation is a valuable research and design tool, the time and difficulty required to create new simulations (or re-use existing simulations) often limits their application. This paper describes the design of the software architecture for the Reconfigurable Flight Simulator (RFS), which provides a robust simulation framework that allows the simulator to fulfill multiple research and development goals. The core of the architecture provides the interface standards for simulation components, registers and initializes components, and handles the communication between simulation components. The simulation components are each a pre-compiled library 'plug-in' module. This modularity allows independent development and sharing of individual simulation components. Additional interfaces can be provided through the use of Object Data/Method Extensions (OD/ME). RFS provides a programmable run-time environment for real-time access and manipulation, and has networking capabilities using the High Level Architecture (HLA).

INTRODUCTION

Simulation is an integral part of aerospace research and design. Its ability to predict complex system behavior makes it valuable to the analysis and testing of many entities, including vehicles, on-board components such as avionics systems, pilot-interactive systems such as cockpit displays, flight control systems, and procedures for operation of vehicles[1].

Simulation can fit into all stages of research and design. During basic research and conceptual design, low- and medium-fidelity simulations can highlight fundamental issues and constraints on system design. As the design progresses, higher-fidelity models can be added to the system so that its output provides increasingly detailed and accurate feedback to designers. At the end of design, high-fidelity simulations can serve as a complement to (or replacement for) flight tests[2], and can serve to train the pilots who will first fly the aircraft.

However, despite the theoretical utility of simulations throughout design, the time and resources required to develop simulation software for research and design projects (henceforth referred to as design projects) can sometimes limit or prohibit their use. The development of simulation software can take a great deal of time and resources, to the extent that 'rapid' development has been described as that achievable in weeks to months[3]. Likewise, the development of simulations from scratch requires personnel with substantial skills in many areas, including software engineering and computer programming, computer graphics, and dynamic modeling.

Lacking the resources to develop a tailor-made simulation (and can not bide the delay it would cause), design projects commonly re-use already-existing simulation software[4]. Existing components may be modified, and existing simulations may have new components added to provide new functionality.

However, the impact of these modifications is often to make the software unusable for future projects or for subsequent stages of the design. Unlike 'traditional' simulation projects (which are focused on providing simulation software of high quality), design projects are often not motivated to maintain software standards, with result that the software quality can degrade. This concept is shown notionally in Figure 1.

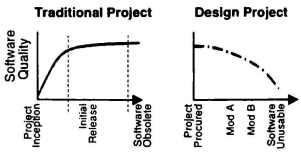


Figure 1. Time History of Simulation Software Quality in 'Traditional' Software Projects, v.s. Design Projects Where Simulation Is a Tool

---

* Senior Software Engineer, Center for Advanced Aviation System Design (CAASD)

† Assistant Professor, Schools of Industrial and Systems Engineering and Aerospace Engineering, Member AIAA

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

Currently, there is no widely available architecture for simulations intended specifically for a comprehensive range of aerospace research and design needs. Solutions are often oriented towards a single objective and are difficult to apply or modify to fit additional requirements; for example, pilot-in-the-loop simulators can be difficult to convert to simulations capable of running fast-time analyses of flight control systems, and vice versa. Solutions that do provide a wide-ranging and robust framework are proprietary, expensive, platform-specific or difficult to obtain.[5]

The remainder of this paper discusses the design of simulation software meeting the needs of aerospace researchers and designers. First, these needs are reviewed. Then, object-oriented programming (OOP) and Object-Oriented Analysis and Design (OOAD) principles are introduced as mechanisms for developing a suitable framework. Based on this discussion, the development of the Reconfigurable Flight Simulator (RFS) is outlined.

REQUIREMENTS OF SIMULATION FOR RESEARCH AND DESIGN

Simulation may be applied in many different ways in aerospace research and design. Conceptual vehicle design may use lower fidelity simulations of single-aircraft flight dynamics as a tool to analyze vehicle sizing and configuration; human factors experiments may use mid-fidelity pilot-in-the-loop simulations; avionics prototyping and design may use simulation as a hardware-in-the-loop testing mechanism; design of operational procedures or mission scenarios may use batch simulations of multiple vehicles; flight control design may use fast-time simulations with high fidelity vehicle dynamic models; and flight tests require a flight simulator with fidelity high in every measure.

This range of needs highlights the need for a design project to have a simulation tool available that is *flexible*; i.e. the simulation should not be fundamentally constrained by its basic architecture to one mode of operation or to one level of fidelity.

A simulation for design projects must also accommodate user types that are fundamentally different than the users of current flight simulators. The first is that of the *general user*, who wants to use the simulation as part of his or her day-to-day design activities. The general user can be defined as a designer who does not want to interact with source code or recompile software. While all simulations must support general users, design projects are novel in that their general user is quite knowledgeable, and desires considerable power over the simulation so that he or she can reconfigure it and use it in a number of ways. With normal flight simulations, the general user is shielded from the underlying functionality through graphical user interfaces or simple configuration scripts[6]; in simulations for design projects, the general user benefits from greater power over the simulation.

Unlike most simulation software packages, simulations for design projects must provide support for an additional user class – that of the *developer*. In traditional software projects, developers are normally dedicated programmers and software engineers, whose primary purpose is to program the simulation. In design projects, the developer is often a member of the design team with some programming knowledge; the developer's purpose is to get the simulation running as a tool. In this case, the developer is motivated to get the simulation running quickly without any supervision of software quality. He or she may not be in a situation to understand the complete workings of all the simulation components, and therefore may not understand the impact of their modifications. Additionally, the developers in design projects may be distributed throughout an organization.

As such, simulation software for design projects must be designed to be inherently *robust* and *programmable:* robust in that modifications to one part of the software should not have widespread, unanticipated effects on other parts of the software; programmable in that the developer should find the overall framework of the simulation easy to understand, and he or she should quickly be able to find where and how modifications should be made to evoke the desired behavior.

To support design projects, simulations must be inherently *extendable*; i.e. it should be easy to add new functionality to the simulator through the addition of new components, rather than through fundamental changes to the entire architecture. Likewise, the components should be maintained in a form such that new components can be added to the simulation without destroying currently existing functionality.

Finally, a simulation will be the most suitable for wide-spread use if several practical considerations are met. For example, many design projects do not need simulations of such high-fidelity that they can not be run on Pentium-based personal computers, while other simulations may be so computationally expensive as to warrant investment in high-speed computers. As such, simulations that are restricted to one platform inherently limit their distribution. Likewise, simulations have more general utility when they do not require specific external hardware or peripherals.

OBJECT-ORIENTED ANALYSIS AND DESIGN

As programs have grown in complexity, Object-Oriented Programming (OOP) has been proposed to create software easy to design, modify and re-use[5]. OOP principles include: abstraction, inheritance, layering, encapsulation, and polymorphism.

Abstraction refers to the ability of OOP to define computational structures not as a sequence of logical functions, but as independent objects, each containing the functions (methods) and data required to perform the functions of that object. With this ability, the software engineer's design process relies on abstracting the overall behavior into individual objects, and then determining their functioning. Objects may themselves contain an inner hierarchical structure of lower-level objects.

Abstraction allows for the entire program to be viewed at the level of abstraction appropriate for the task. For example, a flight simulation architecture may, at a high-level of abstraction, be viewed as a collection of input-output objects, vehicle objects, and a timer object; conversely, a designer interested only in the lower-level task of improving the fidelity of a vehicle module can focus on that vehicle object alone, without needing to see the structure of the rest of the software.

Abstraction also provides a mechanism to structure the software using the same abstractions that are used by other domains. For example, an aircraft object could be broken down into sub-components in many different ways; a common choice is to mirror the different on-board systems (engines, flight controls, etc.). Various works in the literature have described suitable object definitions for flight simulators[7].

Inheritance refers to the ability, using OOP, to define base interface standards for the behavior of a type of object. For example, a base or parent 'aircraft' object might define what functions any aircraft object must be able to perform, and how its data can be accessed. Modules capable of simulating specific aircraft can then inherit from this base aircraft, providing specialization and additional functionality while guaranteeing that these modules will interact with the rest of the program like any object of type aircraft.

Once objects have been defined, OOP's capacity for *layering* can be applied; i.e. smaller, more primitive types of objects can be combined to create larger, more sophisticated objects. For example, an airport type might be made from a combination of runway, tower, hangar and taxiway objects.

Encapsulation is a mechanism by which low-level details about an object are 'hidden' within that object type, so that the entire program does not need to have access to, or work on, the low-level variables internal to an object. For example, an aircraft object type might be required to provide other objects with useful variables such as position, velocity, etc., while variables useful only to the aircraft dynamic model (such as stability derivatives) are encapsulated to stay within the aircraft type.

Polymorphism is an OOP mechanism by which objects inheriting from a parent class can add new functionality while still meeting the base specifications required of the parent class. This feature allows for objects inheriting from the same type to be used interchangeably, a mechanism for code re-use and for rapid reconfiguration.

With the development of these mechanisms, OOP was intended to support software re-use and reduce development costs[8]. However, it was subsequently found that, without additional guidance, OOP can result in unworkable software. For example, introduction of C++ into industry exposed major problems. The first truly large-scale project using C++ and OOP was undertaken in 1988 at Mentor Graphics with the decision to completely redesign their CAD application. The project missed its March 1990 deadline by a year. Beta testing sites reported unusually large numbers of errors, and programmers found it difficult to maintain and correct the code.[9]

Object-Oriented Analysis and Design (OOAD) guidelines and principles have subsequently been developed and tested. OOAD does not add additional mechanisms to those provided by OOP; instead, it specifies effective uses of those mechanisms.

One principle of OOAD is the reduction of overall software complexity. This objective can be achieved by reducing extraneous complexity, and by balancing the complexity. For example, an OOP application could theoretically have many simple objects each capable of only one function, or have only one object capable of all their functions; either extreme makes the software appear complex to a developer. OOAD principles require developers to reduce overall complexity in an object-oriented design

Another OOAD principle is the elimination of cyclic dependencies. An example of cyclic dependencies is shown in Figure 2; three objects all have the capability to call each other's methods. In such a situation, the abstraction of having higher- and lower-level objects breaks down, and other objects in the simulation can not be certain whom to contact to get data from, or give data to, the entire aircraft. To a developer new to the simulator, understanding this implementation can be very difficult.

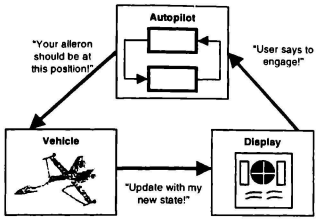


Figure 2. Illustration of cyclic dependencies

Other OOAD principles advise avoiding intrinsically coupled objects. With encapsulation, objects should keep their internal functioning private. However, poorly-designed objects make many of their methods and variables public; in such cases, objects can involve themselves in the internal functioning of other objects. Such internal couplings can result in the violation of abstractions about the functions of objects; as such, modifications to one object might affect a different object in a manner not foreseeable by a programmer. Such situations make the software hard to understand because the control logic for a single process can jump from object to object, and makes testing very difficult because objects cannot be tested independently.

Objects will need to depend on other objects for data. For example, a specialized engine display may require data that can only be provided by a specialized engine model. However, these dependencies can proliferate to such an extent that the use of objects is not reduced to isolated configurations or combinations, as shown in Figure 3. OOAD guidelines help prevent such a proliferation.

In summary, OOP can provide the much needed benefits of understandable, readily modified, easily re-used software. However, these benefits can only be fully realized if OOAD principles and guidelines are followed. These OOAD stipulations can not be met through low-level programming standards that specify only such items as conventions for naming variables – instead, they require the software architecture to be well-thought-out from its inception.

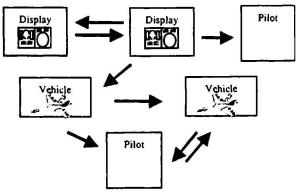


Figure 3. Proliferation of Dependencies Between Objects

DESIGN OF THE RECONFIGURABLE FLIGHT SIMULATOR

The Reconfigurable Flight Simulator (RFS) was designed to meet the requirements of a simulator useful for aerospace research and design activities. The architecture was designed in an object-oriented manner with attention to OOAD principles. It is programmed in C++, with graphics capability provided by OpenGL, to facilitate portability across platforms.

Overview of the Simulator Architecture

The RFS is highly modular. The main RFS application does not contain any simulation models. Instead, as shown schematically in Figure 4, the main application provides the run-time support for individual simulation components. This run-time support includes initializing and registering the individual components, and providing communication between them. The main application also contains the interface standards that the components must inherit from.

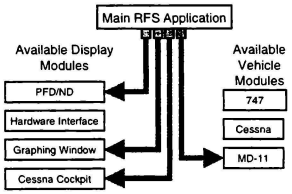


Figure 4. Simulation Access of Components

The components, or plug-in modules, are each stored in a precompiled library that can be loaded by the simulator during run-time. The user can select from a library of available components to configure the simulation as desired for any particular run. Developers can extend the capabilities of the simulator by creating new modules.

The major components of the RFS are shown in Figure 5. Arrows in this diagram represent the access each component has to other objects in the simulation. Cyclic dependencies were avoided by

creating a hierarchy within the components; in the dependencies shown, for example, the scenario object can call the four types of objects that are 'underneath' it, the simulation controller objects can call three types of objects, and so on, with the Environmental Controller and Database (ECAD) object at the bottom of the chain. All of the objects are based on a simulation foundation class (SFC). This is represented in the dependency diagram shown in Figure 6.

To provide a dynamic framework, the RFS architecture supports swapping, removing, and loading components in the simulation during run-time. To prevent problems with violation errors (where objects attempt to access components that have been removed) and to ensure that a newly-loaded component is used properly by other objects, a notification system notifies all components when discrete changes occur.

Components Within RFS

The components of RFS are listed in Table 1, with a brief description of their functions within the simulation. Some of these components provide the functionality within the main RFS application; others establish the standard interface for plug-in modules.

The component-based design of the RFS allows developers to create plug-in modules providing new vehicle models, displays or simulation controllers. Because these components are stored in linked-lists in the simulation, the number of each which can be included in a simulator configuration at one time is limited only by the computer hardware on which the simulator is being run. The functionality of the main RFS application can also be extended.

This component-based architecture has several advantages. Developers in different locations and on different projects can create new components and upload them to a central repository; as such, a distributed development environment is possible. Since each module is encapsulated, the developer can work on individual modules without needing knowledge about other components. This facilitates code re-use and reducing the amount of time to tailor the simulation to particular applications. In addition, simulation developers do not require a broad range of expertise. For example, a flight controls designer need only modify an aircraft's dynamic model and flight control design, using established display modules without needing knowledge of computer graphics.

General users also benefit from this architecture, as they can create new simulations from this growing library of modules. Also, a simulation user can store only those components that are needed for their applications, reducing storage requirements.

This paper has focused on how these components are fit into the RFS framework. The specific details of their underlying models are already well covered in the existing literature, ranging from aircraft dynamic models[10] to computer graphics [for example, 11,12]. Also, graphical programming tools, and code generators are increasingly prevalent for these types of components [for example,13,14].

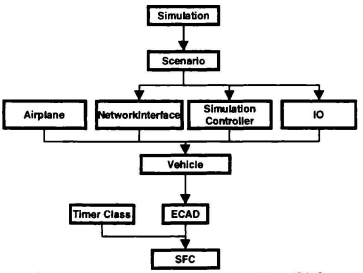


Figure 6. Dependency Diagram of the RFS Standard Interfaces

Object Data/Method Extensions

The base objects within the main RFS application define the minimum communication standards for components within the simulation. However, if these standards were the only communication mechanism, then they would constrain the components unnecessarily. For example, a particularly intricate display may require information from an aircraft that is not available through the base vehicle interface.

To extend these base vehicle definitions, the Object Data/ Method Extensions (OD/ME) interface was created. It establishes a generic, simulation-wide mechanism for message and data passing between objects.

All objects in the RFS have OD/ME capabilities built into their base classes, giving them the ability to declare methods and variables to the entire simulation. The variables can be write-able or write-protected. Other components can then access these methods and variables through the OD/ME interface.

OD/ME adds to the simulation the power to use components with arbitrary communication requirements; this corresponds to an ability to include components of arbitrarily high fidelity and detail. However, because it passes through an interpreter, OD/ME access can be an order of magnitude slower than direct access through the base class interfaces.

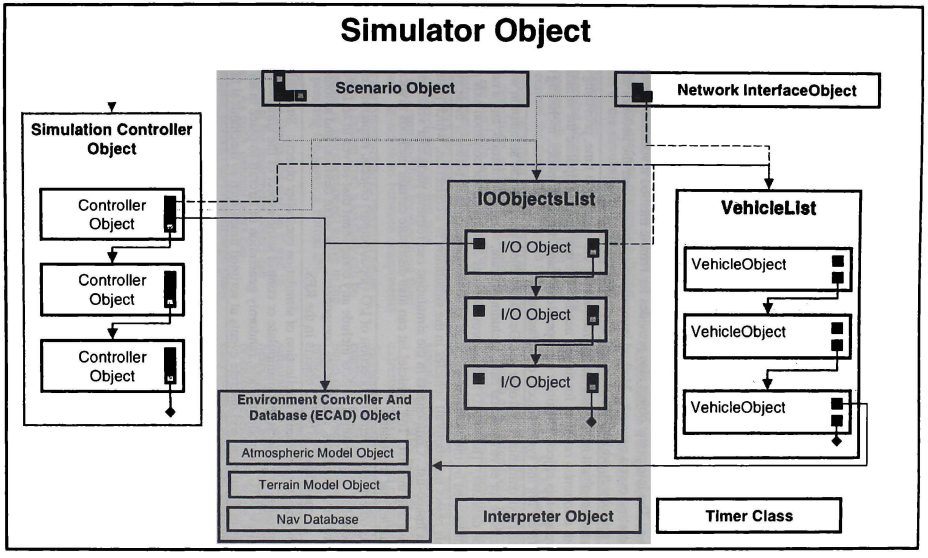


Figure 5. Main RFS Components
(Arrows Represent Access to Other Objects In the Simulation)

Table 1. Description of Core Simulator Objects

Components Integral to the Main RFS Application	
SimulationObject	This object is the main object in the simulation. The SimuationObject maintains all other objects that will take part in a simulation, which entails establishing communication links between objects in the simulation, and ensuring that all objects are notified when an object is destroyed or is no longer taking part in the simulation. The simulation object is also responsible for implementing the main simulation loop, notifying each component and providing them with the new simulation time each time through the loop.
ScenarioObject	The scenario object maintains the simulation modules. It is responsible for loading and unloading DLL modules and extracting objects from the modules. Once loaded, the scenario object is responsible for placing these objects in their appropriate location.
InterpreterObject	The interpreter object is responsible for providing users with the ability to manipulate objects at the method/attribute level in real-time. This object translates user commands into method calls and attribute queries, accepting text strings that are in a format similar to C++. The interpreter object also provides for remote method invocation when RFS is operating over a network. Method invocations are passed over the network and received by this object, which interprets and dispatches the invocation.
TimerObject	The timer object is responsible for maintaining the simulation time, providing components with a common resource for timing needs. The user can manipulate the simulation time through this object. The timer can operate in many different modes, including real-time by referencing the system clock, or in fast-time.
ECAD	The Environmental Controller and Database object provides the simulation with a database for environmental data, including terrain, wind information, and a navigational aide database. The ECAD further provides axis definitions for common axis systems that will be used by the simulation as well as conversion routines between these systems.
VehicleList	This object maintains all vehicles in the simulation in a collection, which is implemented as a linked-list. Components in the simulation can retrieve particular vehicles by traversing this list. The VehicleList can also distribute certain methods invocations to all vehicles that it contains.
IOList	This object implements a collection of I/O (input/output) objects, implemented as a linked-list. Input/output objects include any component that serves in an I/O capacity. Hardware interfaces, data storage components, pilot input devices, and avionics displays are all considered to be I/O objects in the RFS.
ControllerList	This object implements a collection of simulation controller objects. Simulation controllers are objects that manipulate or control some aspect of the simulation. Examples include ATC controller models, randomly generation of aircraft to place in an air traffic control simulation, and discrete events at scripted times such as mechanical failures.
Base Interface Standards for Components to be Added by Developers	
BaseVehicle	This interface object defines communication for vehicles in the simulation. All vehicles in the simulation, including airplanes and ground vehicles, implement this interface. Components can access common parameters such as position, orientation, and velocity. A baseAircraft interface standard has also been created that inherits from this interface.
BaseIO	This interface defines communication for I/O (input/output) objects. This interface comprises mostly of callbacks invoked when certain simulation events occur (such as starting and stopping the simulation, or the start of each time step).
BaseController	This interface defines communication for simulation controller objects, comprised mostly of virtual callback methods. Simulation controllers must implement this interface.

User Control Over Simulation Runs

The OD/ME interface also provides the user with the ability to access all aspects of the simulation during run-time. The most basic method of access is through Graphical User Interfaces (GUIs), such as data viewer windows and graphing windows; such GUIs can be implemented as components meeting the base IO object interface standards.

In addition, the interpreter object can map strings of characters to specific method invocations or data requests. As such, the user can control the simulator directly through text commands in a command-line window. Commands are entered through the console window and parsed by the interpreter. The parsed commands are stored as character strings in the computer memory, which OD/ME can map into method invocations or data retrieval. In this way, the user can manipulate during run-time any object in the simulation through method invocations and data access.

This interpreted access is a powerful feature of RFS. Interpreter commands were built into RFS that allow the user to control the simulation through the command line window, including the ability to add new components to the simulation, and to list all methods and variables that are available within any active component. The interpreter language is similar in form to C++, so that developers and users will not need to learn a new syntax. Developers can test new components from the command line by executing their methods one-by-one. Interactions between components can be configured during run-time to meet the needs of the task; for example, a button from a joystick can be mapped to any method of any component active in the simulator, so that pressing the joystick button can execute the method.

Command line inputs can also be stored in plain text script files that the interpreter object can access and process. As such, general users can configure the simulator before run-time through plain text files, without requiring any access to the source code or recompilation. These script files can also contain references to other script files, so that standard configurations can be defined and referred to, reducing the complexity of any individual script file.

Networking

Networking has proven invaluable for performing simulations when the processing load is too great for a single processor, as is common in flight simulations with high-fidelity dynamic models and/or extensive graphics. Networking is an integral part of many contemporary architectures. For example, the LaSRS++ simulator requires that all components access shared memory, which can be passed between machines running the simulation. However, as is common in current simulation architectures, the networking is at a 'lower-level' than the vehicle, making the vehicle object dependent on the networking object. As a result, changes in networking methods or configuration may require modifications to all components that use networking.

The RFS architecture opted to place the networking a higher level then the components, with the result that networking capabilities are transparent to component developers and components do not need to build in special features to access the networking interface (Instead, the burden is placed on the developer of a networking interface to determine the information that is to be passed). Further, the overall architecture is not reliant on a single networking protocol. Currently, a networking component has been developed that uses the High Level Architecture (HLA) standards, but other networking protocols can be created and used.

CONCLUSION

This paper examined simulation as a general research and design tool. While simulation can bring powerful analysis capabilities to research and design, its use has traditionally been limited by the time and resources required it, and by the use of different simulation architectures in different domains.

The RFS provides a framework that is extremely flexible and extendable to fulfill these needs. Simulation users are provided with near-complete control over the simulation and its components, and can build on existing components to rapidly create new prototypes and solutions.

Through adherence to OOAD principles, the simulation architecture can benefit developers by helping them avoid the common pitfalls of object oriented development while delivering a system that is easy to understand, use, and adapt for a particular purpose. Further, the ability to run on desktop machines without specialized and expensive hardware provides instant accessibility to users and developers on a well-known and easy to use platform.

The RFS has proven to be a stable and scalable platform for development. Designers benefit from a library of components that facilitate development, such as real-time monitoring and graphing components, as well as features such as the programmable real-time environment to provide extreme control over all objects in the simulation during run-time. The RFS has been used for a variety of simulations, from fast-time large-

scale simulations of air traffic control[15], to single-aircraft pilot-in-the-loop studies of cockpit systems[16].

Through its flexible architecture, the RFS provides a single software simulation solution for the entire design process. As a design matures, it requires a simulation that can adapt to new requirements and higher fidelity. The RFS provides a flexible and scalable framework for early design work which can adapt as more detailed and accurate simulations are required. For conceptual design and exploratory prototyping, component reuse and modification of existing capabilities can provide a fast solution. As the design progresses and requirements grow, the RFS adapts by allowing the development and integration of higher fidelity components, hardware interfaces, and other tools to expand the simulation as requirements grow, such as HLA networking to distribute processing load to multiple processors.

The availability of simulation architectures suitable for a wide-range of research and design activities (such as RFS) can be reasonably predicted to have a dramatic impact on aerospace research. Research practices can benefit from a simulation that allows for different researchers to share components and test each other's conclusions. Likewise, the general availability of a simulation may enable small research groups to apply better simulation tools then previously possible.

Such simulation architectures may also have a dramatic impact on aerospace design practices. Designers' needs in a simulation mature as the design proceeds[3,17]. A simulation that can progress in maturity and be used in every stage of the design can serve not only as an analysis tool, but as the repository of design knowledge, to which designers add increasingly detailed specifications in the form of simulation components.

ACKNOWLEDGEMENTS

This work was supported by a contract with the Lockheed Martin Corporation, David Hansen acting as technical monitor, and by NASA grant NAG 1-2175 from the Langley Research Center, Paul Schutte acting as technical monitor. Thanks is also due to the other programmers who have contributed to the development of RFS modules.

REFERENCES

[1] Tuohy, S.T. "An Integrated Vehicle Simulation For the Development and Validation of a Commercial Reusable Launch Vehicle" *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Boston MA.

[2] Totah, J.J. and Kinney, D.J. (1998) "Simulating Conceptual and Developmental Aircraft" *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Boston MA.

[3] Norlin, K.A. (1995) "Flight Simulation Software at NASA Dryden Flight Research Center" *AIAA Modeling and Simulation Technologies Conference and Exhibit.*

[4] Walker, W.H. and Kircher, R.C. (1997) "Modern Component Methods for Air Transport Engineering Simulation Software" *AIAA Modeling and Simulation Technologies Conference and Exhibit*, New Orleans LA.

[5] Leslie, R.A., et al (1998) "LaRS++ An Object-Oriented Framework for Real-Time Simulation of Aircraft" *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Boston MA.

[6] Khoshafian, S. and Abnous, R. (1995) *Object Orientation: Concepts, Analysis & Design, Languages, Databases, Graphical User Interfaces, Standards*, Wiley, New York NY.

[7] Paterson, D.J. (1998) "Simulation Middleware Object Classes (SMOC) for Simulation and Modeling Applications" *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Boston MA.

[8] Alagic, S. et al (1996) "Object-Oriented Flight Simulator Technology" *AIAA Modeling and Simulation Technologies Conference and Exhibit*, San Diego CA.

[9] Lakos, J. (1996) *Large Scale C++ Software Development*, Addison-Wesley, Reading MA

[10] Stevens, B. and Lewis,F. (1992) *Aircraft Control and Simulation*, Wiley, New York NY

[11] Foley, J. et al. (1995) *Computer Graphics: Principles and Practice*, Addison-Wesley, Reading MA

[12] Woo, M. et al. (1997) *OpenGL 1.2 Programmers Guide*, Addison-Wesley, Reading, MA.

[13] Ippolito, C.A. and Pritchett, A.R. "SABO: A Self-Assembling Architecture for Complex System Simulation", *The 38th AIAA Aerospace Sciences Meeting and Exhibit*, Reno NV.

[14] Robins, A. et al (1998) "Commercial Visual Programming Environments: One Step Closer to Real Simulation Reuse" *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Boston MA.

[15] Pritchett, A.R., Lee, S.M., Huang, D. and Goldsman, D. (2000) "Hybrid-System Simulation for National Airspace System Safety Analysis", *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Denver CO.

[16] Pritchett, A.R. and Yankosky, L.J. (2000) "Pilot Performance at New ATM Operations: Maintaining In-Trail Separation and Arrival Sequencing", *AIAA Guidance, Navigation and Control Conference*, Denver CO.
[17] Williams, G.B. (1997) "The Boeing Commercial Airplane Engineering Simulation, New Airplane Project, Management Observations" *AIAA Modeling and Simulation Technologies Conference and Exhibit*, New Orleans LA.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37286

AIAA-2000-4503

DUECA - DATA-DRIVEN ACTIVATION IN DISTRIBUTED REAL-TIME COMPUTATION

M.M. (René) van Paassen* and Olaf Stroosma†
Delft University of Technology, PO box 5058, 2600 GB Delft, The Netherlands
J. Delatour‡
LAAS-CNRS 7, avenue du Colonel Roche 31077, Toulouse Cedex 4

Experiments and flight simulation programs for a research flight simulator can be considered as a combination of real-time distributed calculation processes. Current tools and architectures for implementing these processes are complicated to use and require considerable real-time programming skills. The created programs are not sufficiently flexible. A new middleware layer, DUECA, was developed to facilitate implementation of programs on a research flight simulator. It combines a publish and subscribe mechanism and message passing facilities with some novel elements, namely explicit allocation of process activation and the conditions under which that allocation should take place, synchronisation of data from different sources, and a mechanism to transparently combine processes running at different update rates.

Introduction

THE Delft University of Technology makes use of flight simulation for research, and the research of flight simulation techniques. The foundation of the international institute for research into SImulation, MOtion, and NAvigation, and the construction of the SIMONA Research flight Simulator (SRS) are prime examples of this activity.

Version 2 of the SRS will become operational in May of 2001. This version will contain an elaborate display system and an advanced control loading system. The SIMONA simulator consists of a light-weight carbon and aramid-fiber aircraft cabin mounted upon a high-performance six degree of freedom motion platform (Figure 1).

The low mass and the stiff construction of the cabin, together with the high motion bandwidth of 15 Hz enable high quality simulations in which the pilot will not be aware of any delays between his input signals and the response of the cabin. Such a high frequency response requires a high update rate

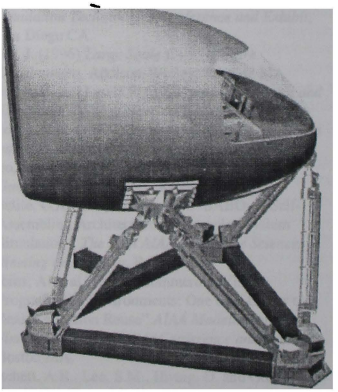


Fig. 1 SIMONA Research Simulator

*Assistant Professor, Faculty of Aerospace Engineering
†Research assistant, SIMONA Research Institute
‡Doctorate Student, Laboratoire d'Analyse et d'Architecture des Systèmes

Copyright © 2000 by the Delft University of Technology. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

and near zero delay of the signals controlling the platform. This update rate can be produced by a real-time computer system especially developed for SIMONA.

Additional information on SIMONA can be found at the following location: `http://www.simona.tudelft.nl`.

The computational infrastructure for the SRS consists of several computers, linked by a high speed communication network (SCRAMNet [1]). This network provides a fiber-optic link between the computers, a global memory space accessible to all computers and interrupt facilities.

A simulation or experiment can be seen as a real-time distributed calculation process, and the following functions can be identified in this process:

- Synchronization of calculation processes with wall clock time
- Communication of data between processes; this communication may cross nodes in the real-time network.
- Matching of calculation processes with communication; i.e. the prevention of race conditions.
- Communication with the simulator hardware.

An experiment with the SRS hardware will require running synchronized real-time programs in several of the computers, and, without advanced programming aids, development of such an experiment is a task that will require considerable expertise on the field of real-time programming and computer communication.

Design and implementation of real-time programs on distributed hardware is normally the work of experts. Especially the design of a communication and activation schedule, and testing the performance of a real-time process is a complicated job. However, the users of the SRS will be scientists and students interested in solving a problem in simulation or human-machine interaction, and are usually not experts in real-time distributed programming. A software environment was therefore needed that supports these users in the development of real-time experiments and simulation. The planned environment consists of two parts:

- A middleware layer, DUECA (Delft University Environment for Communication and Activation). This layer facilitates the development of real-time distributed processes, by hiding the distributed nature of the network, and by providing synchronization of calculating processes.
- An application framework, DUSIME (Delft University SIMulation Environment). DUSIME provides basic capabilities needed for a real-time simulation, such as run control facilities (start/stop, etc.), initial condition calculation support, snapshot taking and logging facilities.

Since experiment development and modification will be a more or less continuing process in SIMONA – as with any research simulation facility – the simulation programs and experiments will be subject to frequent change, modification and extension. DUECA and DUSIME are designed to support rapid prototyping and extension and/or modification of simulation programs.

Existing approaches

Some existing approaches for the implementation of real-time simulations on distributed computing hardware were reviewed:

- The "traditional" use of a common data area (sometimes called data dictionary) that is globally accessible, and a schedule or calling sequence for the calculations, e.g.[1] A drawback of this approach is that schedules for distributed, multi-node calculations are difficult to implement, and very inflexible. A careful design of the schedule, based on data use and production of all calculation processes, must be combined with hand-programmed semaphores (or other synchronization mechanisms) to guard against race conditions. Errors in the design of the schedule are easy to make, leading to time-related imperfections in the implementation of the model. The advantages of this approach is that it has little run-time overhead.
- The use of message passing mechanisms. In this approach the scheduling of processes in the system is triggered by the arrival of data. This eliminates the necessity of a (hand crafted) schedule, and it solves the problem of race conditions. In a traditional message passing application (e.g. with the tools supplied by QNX[3]), the destination of messages has to be known to the sender. Disadvantages of this approach are that the run-time overhead is larger, and the implementation is still not very flexible, e.g. additional processes require modification of the code that sends out data. A more important disadvantage is the programming overhead of the model. Calculation processes for simulation experiments usually require data from multiple sources. In this solution the application

[1] Reg. Trademark of Systran Corp.

programmer has to provide the code that *synchronizes* data with the data coming in from other sources. The advantage is that the schedule is generated by the data communication.

- Message passing in combination with a publish-subscribe mechanism.[6] Examples are the use of real-time Corba,[5] or HLA for distributed simulation.[2] This makes the message-passing solution more flexible. A disadvantage is that the programming overhead is increased.

None of these approaches satisfied our needs to the desired extent. Therefore it was decided to develop a new middleware layer, DUECA, short for Delft University Environment for Communication and Activation.

DUECA

DUECA is specifically designed for the implementation of simulations and simulator experiments. It uses a publish-subscribe mechanism to set up the communication, by which it supports a modular and flexible program design. In addition, the following functions have been added, with the aim of facilitating the implementation of a distributed real-time computational processes:

- Activation is allocated explicitly in DUECA. In a normal message passing system, activation occurs for each incoming message individually. In DUECA activation can be coupled to a combination of messages, so one can specify that a module in the simulation is only activated when *all* inputs needed by that module are available.
- DUECA can wait for a synchronous set of data, and only then activate a process. Even when incoming messages run "out of synchronization", e.g. data from different times is simultaneously available, DUECA can provide a time-consistent set of messages to a user's module.
- Mechanisms are provided to transparently work with data that has different update rates.

This provides a flexible environment for the implementation of simulations and experiments. The simulation can be programmed as a set of modules connected via data channels, in a data-flow architecture. A module in DUECA will be written as a (C++) class, and the steps typically needed at the creation of a module are:

- Publication and subscription is done to obtain a connection to the output and input data channels.

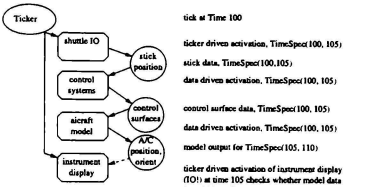


Fig. 2 Illustration of the data-driven activation in DUECA. Three processes are time driven (by the Ticker), input, output and atmosphere calculations. Other processes are activated by the (possibly joint) availability of data.

- An activity object is created.
- This object is linked to a method of the class that implements the activity, i.e. the model calculation.
- The update rate of the activity may be specified.
- The triggering condition for the activity is specified as the simultaneous availability of data on the input channels.

The method that implements the activity is then invoked each time a model update has to be calculated. Upon invocation, this method can access a time-consistent set of data from the input channels, perform the model calculation and send the result over the output channels. By thus hiding the synchronization issues associated with a real-time program, application programmers can use data flow diagram concepts, familiar from control systems engineering and modeling environments such as Matlab/Simulink.

The use of a scripting language (Scheme,[4]) and a simple interface between the scripting language and the modules provides added flexibility and re-use; a simulation or experiment can be composed from existing modules and tailored in a creation script written in Scheme.

Basic concepts

The programming model introduced by DUECA uses the concept of "module" for the components supplied by application developer. A module is a software entity that can use the services supplied by DUECA to communicate with other modules. A number of service levels can be distinguished in DUECA:

DUECA base The base service level in DUECA is accessible to modules written in (currently) C++. This level provides;

1. Naming, identity and registry. Each module using DUECA services must have a unique name. By deriving from the parent NamedObject (usually not directly, but indirectly), this service is provided. In addition to the name, DUECA assigns an Id, composed of a locationId, pointing to the DUECA executable the module lives in and an objectId, which is unique within the DUECA node.

   A pointer to the module (of type NamedObject*) can be obtained by querying DUECA with a name or with a module Id.

2. Communication. DUECA modules can communicate by means of "Channels". A channel is a (possibly) distributed communication device for communicating one specific type of data. Channel versions exist for communicating events and for communicating stream data; i.e. data that is refreshed regularly.

3. Activation. A computational process in DUECA does not require the specification of a global schedule (in the form of: first module A, then module B, then modules C and D, etc.) instead each module describes the conditions under which it has to be scheduled. This can be simple, for example many modules implement a calculation like:

$$x(t+1) = F(x(t), u(t)) \quad (1)$$

$$\text{or:} \quad y(t) = G(x(t), u(t-1)) \quad (2)$$

   For module F, the activation condition is that the data x and the data u are both available at a certain time t. It can then produce data x for time $t+1$. For module G, data x has to be available at time t and data u at time $t-1$, it can then produce data y for time t. For real-time simulation, the activation can also be requested from a "Ticker", which provides activation at regular intervals. See the example in Figure 2.

DUECA configuration DUECA has the capability to implement a computational process in a distributed manner, on a number of computers. The channels in that case will have multiple channel ends, one end in each executable in which that channel's data is read or written. To use DUECA on a distributed platform, one must start a DUECA executable on all the platforms. Each executable must have a unique LocationId, and one must specify how the data is to be transported to the other DUECA parts.

The programming language "Scheme" is used to implement the DUECA configuration. Special scheme commands were added to the interpreter used by DUECA. These commands can be used to set up the transportation media used and to specify which partners get their data by what media. A file "dueca.cnf", in the directory in which the DUECA executable is started, is read and used to obtain configuration data.

DUECA creation DUECA provides an elaborate and flexible system for creating so-called "entities" from modules. An entity is for example an aircraft in a simulation. A DUECA system can contain and update several entities in parallel.

The programming language Scheme is also used to implement the creational process. A scheme script can for example describe the composition of an entity out of several modules. If the modules require or can accept parameters, it is possible to specify these parameters in the script. An example of simple entity is:

```
(define ent
  (make-entity
   "PH-C0Z"
   (list-of-modules 0
    (make-module 'c172-dynamic-model
'initial-position "57.34.18N05.23.06E"
'initial-heading 330)
    (make-module 'c172-ai-pilot))))
```

This would make a Cessna 172[2]. The complete model is programmed in a single module, and it is combined with a module for an artificial intelligence pilot. The plane's initial position is given in a string with earth coordinates, and the initial heading is 330. If the artificial intelligence pilot has any sense she will start taxiing, flying etc., when the simulation is started.

DUECA control This service level can be one of many versions. It provides a human user (op-

[2]Since Scheme is a full-fledged programming language, one could also make procedures that take a name and initial position, and create a Cessna. Add a loop and you can create a whole platform full of Cessna's

erator) the control over the entities in the computational process. The first implementation of this level will probably be an implementation for control of simulations, called DUSIME (Delft University SIMulation Environment). It provides the means to start, stop, take snapshots, replay, etc. Implementation of DUSIME is currently underway.

Activity

In order to play its role in a simulation, a module has two basic needs:

communication In order to perform the model calculations or output to the simulator hardware, a module has to gain access to data produced by other modules, and output data has to be delivered.

activation Especially in a real-time program, the timely execution of the model calculations is important. In general, a module needs to be "called" at a fixed update rate.

In a conventional real-time simulation program, the communication is ensured because parts of the program have access to a common data storage in computer memory, and activation takes place when "entry points", in the form of subroutine calls. In publish-subscribe systems, the communication is provided by a middleware layer. Usually a module requesting data can provide a function (or object method) that gets called when new data is available. The activation is thus triggered by the availability of data.

Allocation of Activity

In the decomposition of a simulation or experiment program, one can identify a number of activities, in which each of these activities consists of a logically coherent set of programming instructions. Some examples of activities are; - the calculation of aerodynamic force and moment contributions of an aircraft model, reading of the primary control inputs, generation of an EFIS display image.

In a conventional simulation program, a number of activities is strung together into a task. In this manner, the start of an activity is triggered by the completion of a previous activity. The problem is that for most activities, completion of a previous activity is *not* an adequate trigger. For most activities, and for all activities involved in updating the model state, the only adequate trigger is the availability of all the necessary input data. In a single-processor, single-thread simulation, one can determine a call

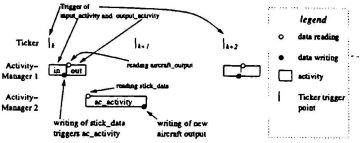


Fig. 3 Sequence of activities for a simple simulation, with time-driven input and output of data, and a data-driven model update.

order in which the completion of the previous activity *coincides* with the availability of the necessary data. For a distributed computation environment, this task quickly becomes complex, while synchronization mechanisms and data buffers need to be introduced.

A module in DUECA may explicitly define one or more activities. An activity can be triggered by various events. One possibility is triggering of an activity on the passing of wall clock time. This is appropriate for activities that implement input or output to the simulator's devices. An activity can also be triggered by the availability of data. If a module uses two or more data sources, it can specify that its activity should be triggered by the simultaneous availability of data from several sources. So the following steps are followed for connecting a module in DUECA:

- Subscribe to all data channels needed for the module.
- Create an Activity, and link it to a method or function that implements the Activities calculations.
- Specify that the Activity should be triggered when the necessary data is available.

The time associated with the data that is being published is stored in the channels. This time stamp is used in the synchronization of data from different sources. As an activity is (re-) started, it is invoked with a time specification as one of its parameters. Using this time specification, the activity can obtain a time-consistent set of data from the channels it reads its data from, and the data produced by the activity can be time-stamped with the same time specification. A simple example of a schedule created by specification of activities triggered by time and activities triggered by data availability is given in Figure 3.

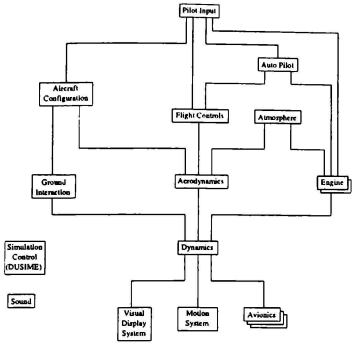


Fig. 4 Basic simulation architecture for an aircraft model in the SIMONA Research Simulator

Example implementation

This section discusses an example implementation of a – fairly basic – aircraft simulation in DUECA. It covers the basic architecture as it is being developed for the SIMONA Research Simulator (SRS). In Figure 4 the different modules and their relationships are shown.

On the top of the figure several modules can be seen which generate input of some sort to the system: the Atmosphere module generates atmospheric properties for use in the Aerodynamics and Engine modules. The Pilot Input controls, possibly through an Auto Pilot, the Flight Controls and the Engine, as well as the Aircraft Configuration, e.g. landing gear and flaps. Since these modules are easily exchangeable, a conventional flight control system with simulated pullies and cables, can, for example, be replaced with an advanced fly-by-wire control system without modifications to the rest of the modules.

The middle section of the figure shows the modules responsible for calculating the aircraft's responses: the Ground Interaction module calculates the forces and moments from the landing gear, but also from other ground contact such as a tail or belly scrape. The Aerodynamics module is responsible for calculating the aerodynamic forces on the aircraft, while the Engine module simulates the engines. All the calculated forces are summed and integrated in the Dynamics module, which outputs the state of the simulated aircraft.

The bottom portion of the figure shows some of the modules which use the aircraft's state in some way: The Visual Display System presents a view of the outside world to the pilot, the Motion System module drives the 6 degree of freedom hydraulic Motion System of the SRS, and the Avionics modules control the cockpit displays. The Sound module collects data from other modules to generate a realistic sound pattern in the cockpit.

The entire simulation is controlled by the Simulation Control module, which is part of DUSIME, the subject of the next chapter.

Extension of the concept: DUSIME

DUECA provides all the tools for implementation of a distributed calculation process. For the implementation of a simulation on DUECA, a number of basic capabilities of simulations must be implemented in all modules, typically a simulation can be stopped, started, reset to an initial condition, data can be recorded, and "snapshots", (complete descriptions of the simulation state) can be taken and used to jump back to a previous point in the simulation.

For the implementation of these properties, common to most simulations, a layer of functionality can be added to DUECA. Different layer can be added when the basic functionality of DUECA is used in other domains, for example data acquisition and identification. The DUSIME (Delft University SIMulation Environment) addition provides the following:

- A class SimulationModule, that provides the coordination between different modules in a simulation, for the purpose of control of the simulation. A DUSIME module would not inherit from Module (the DUECA module class), but from SimulationModule. A SimulationModule knows only two or three basic states:

  HoldCurrent The current state of the model is maintained

  Advance A time step is made for the model

  CalculateInco An initial calculation is being calculated

- A class HardwareModule, that functions as a base class for modules that interface with a simulator's hardware. A HardwareModule has a simple set of states:

  Down The hardware is driven to or kept in the off/safe state

  Neutral The hardware is driven to or kept in a neutral state

Active The hardware uses the model input to determine its output

State transitions between HardwareModules and SimulationModules are co-ordinated, basically the **Active** mode is only possible when the SimulationModules are in **HoldCurrent** or **Advance**.

The simple, orthogonal design of the state diagrams for DUSIME modules simplifies implementation for the application programmer. More complex actions, such as a reset to the initial state of the simulation, will be performed by a HoldCurrent state in combination with a reload from a snapshot, while the hardware is in Neutral or Safe.

A second part of DUSIME is formed by a number of standard modules that provide the coordination of the state transitions, and a user interface for the experiment or simulation controller.

Conclusions

At the SIMONA institute there is a need for a middleware layer that Would facilitate the development of simulations and experiments for a research simulator. Current architectures for the implementation of real-time calculation and communication processes can only be used by experts in real-time programming, and would require too much expertise from SIMONA users. A new design, DUECA, was developed. DUECA combines a publish and subscribe mechanism and message passing with with some novel elements, namely (1) explicit allocation of process activation and (2) the conditions under which that allocation should take place, (3) synchronization of data from different sources, and (4) a mechanism to transparently combine processes running at different update rates.

The implementation of an aircraft simulation in this framework shows that the programming model differs from that used in traditional approaches. Activation of modules, and the flow of control in the program, is determined by the flow of data. The composition of a simulation or experiment from modules, and the extension of a program is however greatly simplified by this mechanism. Not only is the data access facilitated by the publish-subscribe mechanism, timing of activation is also provided by DUECA.

An extension to DUECA, DUSIME, further facilitates the implementation of simulation-specific functionalities.

Acknowledgements

The concept of time specification, the ticker and the manner in which Activity objects are used as a metaphor for thread-of-control in DUECA were born and further developed in discussions between the first and third authors, during a stay of the latter at the Delft University of Technology. This stay was supported by a grant from the Van Gogh Foundation, for which we are truly grateful.

References

[1] Anon. EuroSim mk1 - software user manual. Technical report, Fokker Space B.V., Leiden, The Netherlands, 1997.

[2] Anon. Dmso high level architecture. Webpage, http://hla.dmso.mil/, 2000.

[3] Anon. Www homepage: Qnx software systems ltd. Webpage, http://www.qnx.com/, 2000.

[4] R. Kelsey, W. Clinger, J. Rees, H. Abelson, R. K. Dybvig, C. T. Haynes, G. J. Rozas, N. I. Adams IV, D. P. Friedman, E. Kohlbecker, G. L. Steele JR., D. H. Bartley, R. Halstead, D. Oxley, G. J. Sussman, G. Brooks, C. Hanson, K. M. Pitman, and M. Wand. *Revised(5) Report on the Algorithmic Language Scheme*, 1998.

[5] D. Schmidt. Tao overview. http://www.cs.wustl.edu/ schmidt/TAO-intro.html, Oktober 1999.

[6] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37287

AIAA-2000-4504

RAPID PROTOTYPING OF EFIS SOFTWARE BEYOND FORMATS AND AUTOCODE

E. Theunissen* and R.M. Rademaker†
Delft University of Technology, P.O. Box 5031, 2600GA Delft, The Netherlands
tel: +31 15 278 1792, fax: +31 15 278 1774, e.theunissen@et.tudelft.nl

ABSTRACT

Most rapid prototyping tools support the user with format design by means of a CAD environment and typically generate an implementation through autocode. For conventional electronic flight instrument display formats, such rapid prototyping tools can increase the efficiency of the design process significantly. Spatial display formats such as those used in experimental synthetic vision displays typically combine a presentation of 3-D data with 2-D elements. Although tools exist that support the user with the visualization of 3-D data, integration with the autocode used to render the 2-D instruments and other symbology must be performed at the graphics application program interface level. To allow a seamless cross platform migration of the implementation, transparent support of the desired range of graphics application program interfaces is needed. To achieve this, the focus must go beyond formats and autocode, and should address the software architecture that is needed for a more seamless integration. This paper discusses how ideas with respect to cross-platform migration, maintenance at the specification level and integration have been applied to develop a software architecture that is used for the implementation of hybrid 2-D/3-D display concepts. Furthermore, the approach used for verification and debugging is illustrated.

INTRODUCTION

Today's electronic flight instrument displays typically present a number of indicators to convey information regarding altitude, speed, heading, flight modes, etc. Nowadays, the design of the desired representation of these instruments can be performed with the aid of rapid prototyping tools. Such tools typically comprise:

- A CAD tool that allows the user to design the display format;
- an interface definition tool that supports the user with the specification of the dynamic behavior, and
- a code generation tool that generates an implementation.

Ellis[5] discusses an evaluation of two tools for the design and development of cockpit displays and provides recommendations for the ideal tool. Part of these recommendations address the effort needed for integration.

Several candidate formats of future display formats include some form of spatially integrated data presentation. Examples are the FMS map display developed at Rockwell Collins, the taxi-map display discussed by Andre et al.[2], and various synthetic vision display formats for the primary flight display[1,6]. These display formats typically are a combination of a projection of 3-D data, various symbolic enhancements, and 2-D instruments.

For the development of display formats that depict an artificial presentation of the environment, tools exist that aid in the generation and subsequent visualization of databases from digital terrain elevation data (DTED), digital feature analysis data (DFAD), etc. However, to create a program that generates the desired picture it is necessary to combine the database visualization function of the 3-D environment visualization tool with the code that generates the instruments and symbolic

---

*Associate Professor, Department of Electrical Engineering, AIAA Member

†Researcher, Department of Electrical Engineering.

[1]Copyright © 2000 by E. Theunissen. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

enhancements, which is either hand-written or produced with a conventional rapid prototyping tool. The level at which the various modules can be integrated is just above the graphics application program interface (API). Fig. 1 illustrates this form of integration.

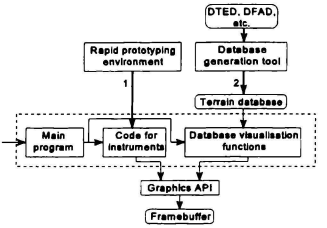


Fig. 1. Block diagram showing the integration of the functionality for 3-D environment depiction together with conventional instruments. The dashed box contains the three elements of the display software. The code for the instruments has been produced with a rapid prototyping environment (1) and the terrain database that is visualized has been created (2) with the aid of a database generation tool.

Examples of this approach are the synthetic vision display discussed by Sachs and Möller[4], and the spatial display discussed by Alter et al.[1]. The first one relies completely on hand-written code for the implementation of the display software and the second one uses the OpenGVS toolkit to render a presentation of the 3-D terrain and additional C-code for the presentation of some flight instruments. As a result of the integration just above the graphics API, the only platforms on which the resulting implementation can be tested are those that support this particular API. This can pose a severe restriction when it becomes necessary to rehost the implementation to other target systems, e.g. flight hardware, which typically have their own proprietary API.

To increase the efficiency of the overall design and evaluation process, a prototyping environment for advanced display concepts should enable the seamless integration of spatially integrated data presentation with conventional symbology and instruments for the full range of target API's that need to be supported. Also, the effort needed to add support for a new target API should be kept to a minimum. In this paper, an approach is discussed that satisfies these requirements. For the past ten years it has been used within the Delft Program for Hybridized Instrumentation and Navigation Systems (DELPHINS) for the development of advanced spatial display formats and their implementation on a wide range of target systems, including flight hardware. Although both format design by means of CAD tools and autocode are elements of the approach, the critical factor is the software architecture of the implementation.

Besides the effort needed for the generation and integration of the functionality, the efficiency of the overall design process depends on the effort needed for verification and debugging. Before the approach to the design of the software architecture is discussed in more detail, the next section provides a brief overview of the various phases of the design process.

THE DESIGN PROCESS

Fig. 2 shows the design-implementation-evaluation cycle and indicates the phases in which rapid prototyping tools are typically used.

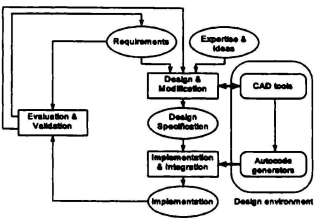


Fig. 2. The iterative design cycle. A design environment comprising tools to capture the design specification and tools to translate the specification into an implementation can significantly increase the efficiency of the design process.

The design process consists of a translation of the requirements into a design specification and a subsequent translation into an implementation. The feedback loop from the implementation to the design and modification phase is needed to further refine the design. Often, initially requirements are not complete, leave room for trade-offs, or may contain undiscovered ambiguities. The feedback loop from implementation to requirements is needed to allow an increase in level of

detail of the requirements. In the ideal situation, the iterative design cycle will rapidly converge to a certain implementation meeting all requirements.

Verification and debugging

Errors can occur both from an incorrect specification of the requirements and an incorrect translation into an implementation. When the implementation does not behave as expected, the designer must determine the cause of the problem. The efficiency of this process is strongly influenced by the effort needed to obtain and compare the required information. To determine the cause of a mismatch between the requirements specification and the implementation, two additional verification cycles are introduced. One cycle to check the specification against the requirements, and one to check the implementation against the specification. Fig. 3 illustrates where these two cycles fit in the overall design process. Given an adequately tested code generator, the likelihood that an error is caused by an incorrect translation is very small, and thus the second verification cycle only applies when it is expected that a mismatch between the specification of the requirements and the implementation is caused by an error in the code generator.

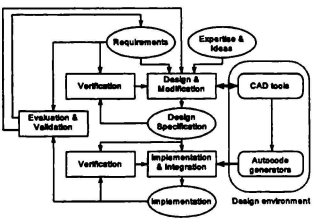


Fig. 3. Adding two verification cycles to the design process. The upper cycle is needed to check the design specification against the requirements, and the lower cycle is needed to check the implementation against the design specification.

GOALS & APPROACH

The main goal is an increase in the efficiency of the overall design and evaluation process. The approach that has been used relies on the application of CAD tools, interface definition tools, and code generation tools but, like mentioned before, the critical factor is the software architecture. This section will focus on the following aspects that have played an important role in the development of the overall DELPHINS software architecture:

- Cross platform migration
- Maintenance at the specification level
- Integration of display objects

As pointed out in the previous section, the efficiency of the design and evaluation cycle is also strongly influenced by the effort needed for verification and debugging. Therefore, the fourth aspect that will be discussed in this section is the support for verification and debugging

Cross platform migration

During the research into display concepts, the implementations that are created represent a significant investment in software. Preferably, a seamless transition of this software to a new graphics system should be supported. To a certain extent this is possible by using a standard API to generate the graphics. However, at a certain point in time, there may be the need to rehost a certain implementation on an actual electronic flight instrument system (EFIS). The API's used with today's EFIS displays typically are different from the standard API's such as OpenGL™. In order to allow the concepts to be migrated to a range of target systems, a display list based 2-D/3-D retained mode internal graphics command set was developed. The implementation generated by the DELPHINS code generator can be divided into the function blocks shown in Fig. 4.

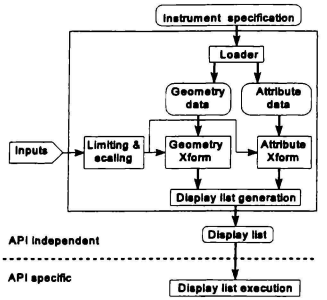


Fig. 4. Block diagram showing how the specification of an instrument is processed and used to generate the representation for a certain state of the input. The blocks within the dashed block comprise the functionality needed to generate the 2-D/3-D instrument object.

As can be seen from Fig. 4, the only API specific part of the code concerns the display list execution.

For various graphics API's, display list execution modules were developed. Table 1 gives an overview of these API's.

Table 1. API's for which a display list execution function has been implemented.

Year	API	Chipsets
1990	TIGA™	TMS34010,34020
1991	BGI	PC EGA,VGA
1995	IRISGL™	Many
1996	MGL™	PC graphics cards
1997	GLIDE™ 1x,2x	Voodoo 1,2™
1998	GLIDE™ 3x	Voodoo 2,3™
1999	OpenGL™ 1.2	Many
1999	CGEI	Collins GE1, GE2

The concept of the display list based 2-D/3-D internal graphics command set is not optimal from a performance point of view since it requires an intermediate step between parameter calculation and command execution. However, the resulting overhead is typically negligible. In the translation of the display list to the graphics API, special hardware capabilities are taken into account in order to optimize performance.

Maintenance at the specification level

As indicated by Balzer[3], one of the fundamental flaws in the existing software paradigm is the fact that maintenance is performed at the implementation level rather than at the specification level. A main goal during the development of the rapid prototyping environment was to ensure that the users would make all changes at the specification level. In the DELPHINS software this has been achieved by ensuring that all relevant parameters that determine the layout of the user interface are not available in the source code but loaded during runtime from the same specification that was used to generate the code. As a result, the only option for the user to make modification to the layout is to work at the specification level. This always ensures consistency between the specification and the implementation

Integration of display objects

Cleaveland[4] indicates that since a code generator is frequently part of a larger system, interfacing the automatically generated code with the rest of the system is very important. The second of Cleaveland's seven steps to building a generator therefore requires the definition of domain boundaries, and an answer to the question of what and where the interfaces should be.

Fig. 4 showed the implementation generated by the DELPHINS code generator. The block diagram in Fig. 5 illustrates an object that is used to integrate the full 3-D capability into the system.

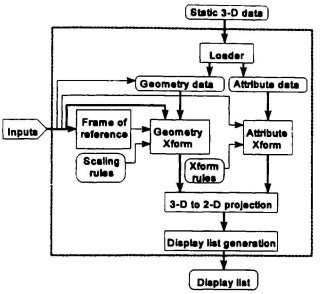


Fig. 5. Block diagram illustrating how a 3-D object generates the display list. To view the 3-D data, the geometry transform either gets data that is loaded into memory, or dynamic geometry data that is provided through the input interface. The position from where the data is viewed is determined by the frame of reference. The scaling rules determine whether a veridical scaling is applied to the 3-D object, or whether some form of geometric enhancement is used.

Both in Fig. 4 and Fig. 5, three interfaces can be identified. The first one is at the top of the block towards the specification of the object. This interface is used during program initialization to load the desired specification of the representation into memory.

The second interface is the input interface. It drives the transforms of geometry and attributes according to a set of rules. For the 2-D/3-D object depicted in Fig. 4, the code that implements these rules is automatically generated from the specification of the dynamic behavior of the object. It is this interface that is used to integrate all objects with the software that must control them.

The third interface is the output from the display list generation at the bottom of the block. In the introduction it was pointed out that an integration of display objects just above the graphics API poses restrictions with respect to rehostability. Fig. 6 illustrates the approach used with the DELPHINS system. The output from the objects as depicted in Fig. 4 and Fig. 5 is routed to a number of selectable display lists. These list are processed by the dislay list execution function. This function is implemented just above the graphics API.

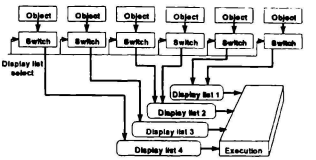


Fig. 6. Block diagram illustrating how the display list select input is used to route the output of the display objects to the display list buffers.

The structure illustrated in Fig. 6 provides the following possibilities:

1. It allows the use of multiple display processors that are configured in an underlay/overlay configuration.
2. It makes it easy to perform parallel execution of the display lists in multiple processor systems.
3. It makes it easy to have multiple windows/viewports on systems that do not support these features in their API.
4. The desired priority among the objects can be achieved by scheduling the objects.

Integration with the rest of the software is performed by connecting the inputs of the 2-D and 3-D objects to the appropriate variables.

Support for verification and debugging

To support the verification process, the representation of the specification and the traceability of the implementation process have been addressed. As can be seen from Fig. 3, the specification is an input to the implementation- and both verification processes.

The instrument specification is generated from the CAD and interface definition tools, and is stored as a readable ASCII text file. The structure of this file is such that it can be understood by a designer who is familiar with the specification language. As was pointed out in the section about maintenance, this file serves both as the input to the code generator and the resulting program. In the latter case it is loaded during runtime to retrieve the specification of the geometry and attributes.

To support the verification of the requirements against the design specification, a graphical representation of the specification can be generated that shows the

relations between inputs, transformations, objects, and attributes by means of a block diagram similar to the one depicted in Fig. 4.

In case it is necessary to verify the implementation against the specification, information can be provided that allows the code generation process to be traced. The actual implementation is derived from the specification via a series of formal manipulations, also referred to as transformations. During the code generation process, a report file is created that contains information about each transformation. Together with the block diagram generated from the same specification, this greatly supports the programmer in determining the relation between each element in the specification and the resulting code that has been generated.

RESULTS

The approach described in this paper served as the basis for the development of the prototyping environment that is being used in the context of the Delft Program for Hybridized Instrumentation and Navigation Systems. The design system described by Theunissen[7] served as the basis for the system discussed in this paper. In a cooperative effort between Rockwell Collins and Delft University of Technology, a display list execution function was developed to support the Collins GE1 and GE2 graphics engines. Fig. 7 shows one of the setups in the laboratory of the radionavigation group at the Delft University of Technology. The top two displays are real flight displays provided by Rockwell Collins. Both the prototyping environment and actual target hardware have been integrated in a single simulation environment.

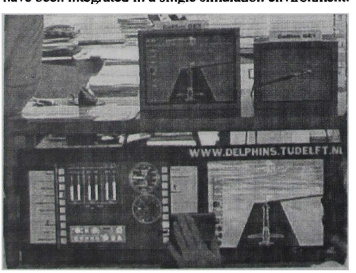


Fig. 7. Display setup showing the research displays at the bottom and the actual flight displays at the top.

For each graphics API, the development system has a specific library that contains the initialization and display list execution functions. The other libraries are common for all API's. Since the system is being used in a research environment, new requirements frequently arise. As a result of the high commonality, new features become immediately available for all target systems and new concepts that are being prototyped on the development system can be tested in parallel with the implementation on real target hardware.

SUMMARY AND CONCLUSIONS

In the introduction it was pointed out that to increase the efficiency of the overall design process for advanced display concepts, it is necessary to go beyond the use of CAD tools for format design and code generation tools for the implementation. This paper has focused on a software architecture that supports the seamless integration of the functionality needed to generate a 3-D picture with the functionality needed for symbology and conventional 2-D instruments, and allows rehosting to a wide range of target systems, including flight hardware. The following elements have contributed to the success of this approach:

- Integration is kept simple by having a single, common input interface to all objects.
- The implementation of these objects does not directly interface to the graphics API.
- The use of a display list architecture is the key to minimize the effort needed for multiple API support.
- The support for 3-D functionality above the internal graphics command set allowed the generation of spatial displays on every available graphics API, including the 2-D ones used with today's EFIS displays.

ACKNOWLEDGEMENTS

The authors would like to thank the Dutch Technology Foundation STW for their support of the DELPHINS II program. Furthermore, thanks go to Rockwell Collins for providing the hardware that allowed the concept to be implemented and tested on a real avionics display system.

REFERENCES

1. **Alter, K.W., Barrows, A.K., Enge, P., Jennings, C.W., Parkinson, B.W. and Powell, J.D.** (1998). 'Inflight Demonstrations of Curved Approaches and Missed Approaches in Mountainous Terrain', *Proceedings of the ION GPS Conference*, Nashville.

2. **Andre, A.D., Hooey, B.L., Foyle, D.C., McCann, R.S.** (1998). 'Field Evaluation of T-NASA: Taxi Navigation and Situation Awareness System', *Proceedings of the 17th DASC*, pp. F23.1-F23.8, Bellevue, WA.

3. **Balzer, R., Cheatham, T.E., Green, C.** (1983). Software Technology in the 1990's: Using a New Paradigm. *IEEE Computer*, Vol. 16, No. 11, pp. 39-45.

4. **Cleaveland, J.C.** (July 1988). Building Application Generators. *IEEE Software*, 25-33.

5. **Ellis, T.D.** (1993). 'An Evaluation of Software Tools for the Design and Development of Cockpit Displays', *Proceedings of the AIAA Flight Simulation Technologies Conference*, pp. 275-280, Monterey, CA.

5. **Sachs, G. and Möller, H.** (1995). 'Synthetic Vision Flight Tests for Precision Approach and Landing', *Proceedings of the AIAA GNC Conference*, August 7-10, Baltimore, MD.

7. **Theunissen, E.** (1993). 'D3S: The DELPHINS Display Design System', *Proceedings of the AIAA Flight Simulation Technologies Conference*, pp. 62-67, Monterey, CA.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37288

AIAA-2000-4574

DEVELOPMENT OF A NEW IN-FLIGHT SIMULATOR MuPAL-α

Kazuya MASUI
National Aerospace Laboratory, 6-13-1 Osawa, Mitaka-shi, Tokyo 181-0015, JAPAN

Yukichi TSUKANO
Japan Science and Technology Corporation
Kawaguchi Center Building 14th Floor, 4-1-8 Honchou, Kawaguchi-shi, Saitama, 332-0012, JAPAN

ABSTRACT

National Aerospace Laboratory of Japan has developed a new in-flight simulator named "MuPAL-α", with the aim of providing an experimental facility for the research and development of advanced aeronautical technologies. MuPAL-α is based on a Dornier 228-200. A fly-by-wire control system and a direct lift control system realize the variable stability and response capability, which is applicable not only to the motion simulation but also to the flight demonstration of advanced guidance and control technologies. The other features of MuPAL-α are its unique second cockpit and data acquisition system with high accuracy and extensity. The second cockpit with a head-mounted display will provide a flexible environment for the research on various pilot interfaces and the effects of visual and motion cues on pilots. In a safety aspect, MuPAL-α adopted the duplex system to improve the probability of failure detection. The auto-disengage function incorporated in the fly-by-wire control system ensures MuPAL-α from exceeding the proven operational envelope of Do228-200. This paper deals with the design requirements, the system architecture and the topics in the development phase.

INTRODUCTION

NAL (National Aerospace Laboratory) had been operating an in-flight simulator, named "VSRA (Variable Stability and Response Airplane)", for about 10 years[1]. VSRA was based on a Beechcraft 65 Queenair and equipped with a fly-by-wire control system and a direct lift control system. VSRA had been used for a variety of experiments, such as evaluation of flying qualities of various types of aircraft, development of gust alleviation control laws[2]. However it has become difficult for its old-fashioned system to catch up with the latest research activities corresponding to the remarkable progress in computer and sensor technologies.

This motivated NAL to develop a new in-flight simulator, which could provide a flight demonstration environment for the research on various themes, such as flying qualities, guidance and control technologies, human factors. NAL named the new in-flight simulator MuPAL (Multi-Purpose Aviation Laboratory). MuPAL consists of MuPAL-α based on a fixed-wing airplane and MuPAL-ε based on a helicopter. "α" and "ε" are the capital letters of the Greek words representing an airplane and a helicopter respectively. In addition to the research themes peculiar to each type of aircraft, the combination of two different types of aircraft will extend the envelope for flight demonstration.

MuPAL-α is based on a twin-engine turboprop commuter airplane, Dornier Do228-200 (fig. 1). MuPAL-α aims at contributing to the development of advanced technologies, mainly relating to aircraft operation, and to the fundamental research in the fields of flight measurement and flight control. As the research on advanced technologies to operate aircraft more safely and efficiently, the flight evaluation on the following themes will be performed, simulating a variety of airplanes, such as big transports, business jets and small airplanes for general aviation;

- a new landing approach path and a guidance system in order to use a terminal area more efficiently and reduce the population suffering the aircraft noises,
- pilot responses against the turbulence, such as gusts and windshears,

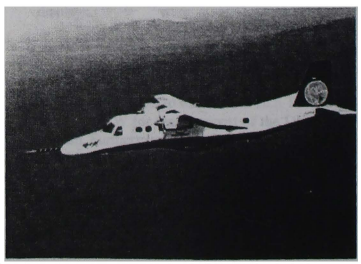


Fig.1 Do228-200 (MuPAL-α)

*Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All right reserved.

- pilot recognition and responses in the event of system failure,
- automatic compensation systems against turbulence and system failures,
- integrated displays and synthetic visions making up for insufficient outside views.

As the fundamental research, MuPAL-α will promote the research activities, such as

- flight demonstration of advanced control theories,
- flight evaluation of flying quality criteria,
- improvement of estimation accuracy of aerodynamic characteristics and comparison with the wind tunnel test data and the CFD calculation,
- observation of winds around an airport,
- flight evaluation of newly developed on-board equipment,
- the research on the effects of visual information and motion cues,
- the feasibility study on the in-flight simulation for a supersonic airplane, spacecraft and rotorcraft.

NAL launched the development of MuPAL-α in 1994 and has obtained the airworthiness certification of Category X (the category for special aircraft including experimental aircraft prescribed by the Japanese Civil Aviation Bureau) in March of 2000. The following chapters present the design requirements, the system architecture and the topics in the development phase.

DESIGN REQUIREMENTS

The major design requirements for MuPAL-α were;

- variable stability and response capability, which enabled the motion simulation of various types of aircraft and the demonstration of advanced control theories,
- pilot interfaces, which could offer a variety of visual information and motion cues,
- sensors with high accuracy and an extensive data acquisition system which was applicable to various types of sensor output signals.

Variable Stability and Response Capability

In order to realize the variable stability and response capability, a FBW (Fly-By-Wire) system and a DLC (Direct Lift Control) system were newly developed for MuPAL-α.

The FBW system was so designed as to extend its flight envelope and maneuverability as close to those of Do228-200 as possible. But the operational limits from structural strength reasons were imposed on DLC system, that is, the maximum speed is 150kt (77m/s) and the load factor has to be kept between 0 g and 2 g (fig.2). This maximum operational speed of DLC

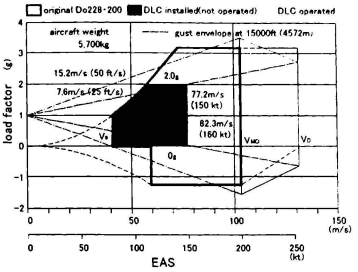


Fig.2 Operational envelope for DLC system (Clean configuration)

system is acceptable because MuPAL-α can simulate the landing approach speed of most of transports and airplanes for general aviation, which means that the outside views during a real landing approach is correctly simulated as an important visual cue for the pilot.

Other design requirements for the performance of FBW and DLC systems were as follows;

- Calculation rate of flight control laws should be 50Hz.
- Frequency response bandwidth of FBW actuators should be greater than 5 Hz.
- Vertical acceleration induced by DLC system should be greater than 0.2g at 100kt (51m/s), which was equivalent to 0.45g at 150kt (77m/s). Frequency response bandwidth of DLC actuator should be greater than 2Hz.
- Flight control laws and navigation and guidance programs could be changed freely by researchers without affecting the flight safety.

The simulation of gust responses considered, the optimal bandwidth of frequency response of DLC actuators should be 5Hz. But, due to the limitation of electric power supply, the requirement was degraded to 2Hz, which is the same as VSRA.

To assure the flight safety, it was required that a safety pilot could take over the control with Do228's original mechanical flight control system whenever necessary. Both the FBW and the DLC systems were designed conservatively to think much of their reliability. The duplex system was required to improve the probability of failure detection and to create a fail-passive system. But the triplex system was not required because of too much weight and power consumption.

Pilot Interfaces

The flight evaluation of various pilot interfaces is one of the most important tasks for MuPAL-α. The pilot interfaces include an integrated display, a synthetic vision, control devices, cockpit arrangement, etc. But the original cockpit (the front cockpit) suffers from many restrictions due to space provision and influences on original instruments. To solve this problem and to provide a flexible research environment for human factor studies, MuPAL-α was required to have an experimental cockpit ("the second cockpit") in its cabin.

A progressive design was required for the second cockpit. The design requirements were as follows;

- A HMD (Head Mounted Display) should be prepared in addition to a flat LCD (Liquid Crystal Display).
- Both the computer-generated images and the real outside view captured by a camera could be presented.
- The time delay of displayed images from aircraft and pilot's head motion should be less than 100ms.
- The contents and formats of display could to be changed freely by researchers.
- The install position of the second cockpit should be changeable so that the different motion cues could be imposed on the pilot.

As for the front cockpit, two sets of 6-inch MFD (Multi-Function Display) or a 10-inch LCD display were required to show the flight conditions of modeled aircraft and the flight guidance.

Data Acquisition

As the sensors for aircraft motion and pilot input are essential to realize the variable stability and response capability, they were required to have not only the high accuracy but also the reliability as a part of the flight control system. To have the applicability to various data, the data acquisition computer was required to have many kinds of interface provisions. The real time data monitoring during flight tests and the quick data processing after flight tests were also required.

The design requirements for the data acquisition system were as follows;

- Accuracy of attitude, angular velocity and acceleration should be better than 0.05deg, 0.05deg/s and 0.02 m/s^2 respectively.
- Accuracy of airspeed and airflow angle should be better than 1kt (0.5m/s) and 0.1deg respectively.
- Data acquisition rate should be more than 600kbit/s.
- Data recording capacity should be more than 3 hr.
- Neither the load factor of 2g nor the low pressure at 25000ft (7620m) should affect the data recording

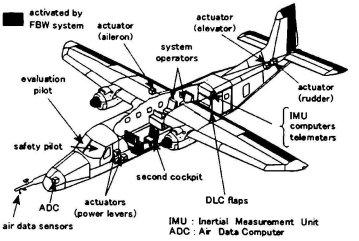


Fig.3 Overview of MuPAL-α

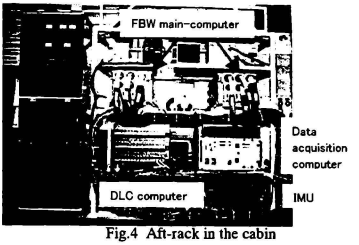


Fig.4 Aft-rack in the cabin

As NAL usually performs the flight tests around the altitude of 10000ft (3048m), the maximum operational altitude of FBW, DLC and second cockpit systems was set at 15000ft (4572m). In order to extend the flight demonstration envelope for on-board equipment as much as possible, the maximum operational altitude of sensors and data acquisition system was set at 25000ft (7620m), which is the same as the operational limit of Do228-200.

ON-BOARD SYSTEM

The FBW system activates Do228's original control surfaces and engine power levers (fig.3). The actuators for elevators, ailerons and a rudder are installed in the same way as the auto-pilot system of Do228. Newly developed DLC flaps enable MuPAL-α to control the three degrees of freedom of longitudinal motion. Two pilots operate MuPAL-α. A safety pilot operates the aircraft by using Do228's original mechanical flight control system. An evaluation pilot, seated in the R/H cockpit seat or at the second cockpit, flies with the FBW and DLC systems. Two or three operators for experimental equipment are seated in the cabin. Computers, telemeters and IMU (Inertial Measurement Unit) are mounted in the back of the cabin (fig.4). The air data sensors are installed on the nose boom.

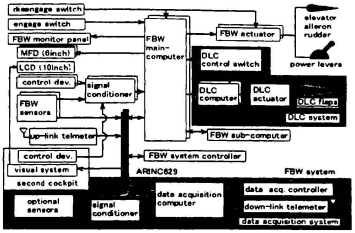


Fig.5 On-board system architecture

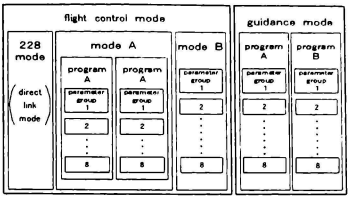


Fig6 Fight control modes and guidance modes

The on-board system of MuPAL-α comprises four sub-systems (fig.5) interconnected by the ARINC629 data bus. The primary components of FBW and DLC systems are duplicated. By installing only the necessary subsystems for each mission, the remaining space and payload can be used for additional equipment.

FBW System

As shown in fig.5, the signals from the engage switch, control devices and various sensors go into the FBW main-computer. The disengage signal goes directly to FBW actuators as well as the FBW main-computer. The FBW main-computer generates the commands for FBW actuators and the DLC system. And it sends the flight conditions and the guidance information to the MFD or the second cockpit. The FBW sub-computer assists the FBW main-computer when necessary. The FBW operator can change the flight control laws by using the FBW system controller. The up-link telemeter receives the data for differential GPS[3] and so on. The status of FBW system is displayed on a FBW monitor panel in the front cockpit.

Sensors

As the IMU, an embedded GPS/INS was adopted because of its high accuracy and lightness of 10 kg. The feature of newly developed ADC (Air Data Computer) is the direct measurement of dynamic pressure by a differential pressure sensor. The functional tests proved its high accuracy of 0.3kt (0.15m/s) even at the airspeed lower than 100kt (51m/s). A conventional vane system with duplex potentiometers was adopted to measure the airflow angle. The vanes were so designed as to be applicable to the gust observation. The RVDT with a duplex pickup is installed to measure the position of each control device and control surface.

As for the case that the installation of a new sensor might affect the flight safety (e.g. engine output torque), the output of Do228's original sensors is divided and transmitted to the FBW system.

Computers

For in-flight simulation, the FBW main-computer calculates the motion of modeled aircraft and makes the plant (Do228) to simulate it. Simultaneously the FBW main-computer outputs the flight conditions of modeled aircraft and the guidance information. The FBW main-computer also performs the system management including failure detection. Because of the adaptability to various types of signals and the reliability as on-board equipment, the FBW main-computer was newly developed as a VME bus computer. On the other hand, the FBW sub-computer is a portable workstation so that it can be replaced easily with more powerful machine in the future. The FBW sub-computer assists the FBW main-computer in the calculation for navigation, guidance and state estimation and so on.

Calculation rate in the FBW main-computer is 50Hz. The calculation of flight control laws and guidance data is allowed to occupy 6ms, which is almost one third of each calculation frame. Three flight control modes and a guidance mode are prepared (fig.6). Except the 228 mode (a direct link mode), the program design is open to researchers. Even if a flight control mode is being engaged, the FBW operator can change the whole or a part of parameter group in order to change the stability and response characteristics.

MuPAL-α aims at simulating the motion around the several trim conditions continuously as well as the simulation around a trim condition, which was already established by NAL's old in-flight simulator VSRA[4]. New control theories will be used to increase the robustness against the uncertainty of the plant model and the measurement noises. As a guidance program for landing approach, the tunnel-in-the-sky display combined with a differential GPS technique is planned[5]

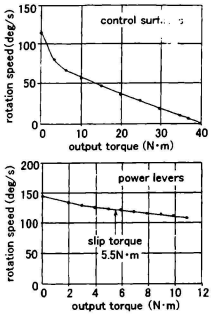


Fig.7 Performance of FBW Actuators

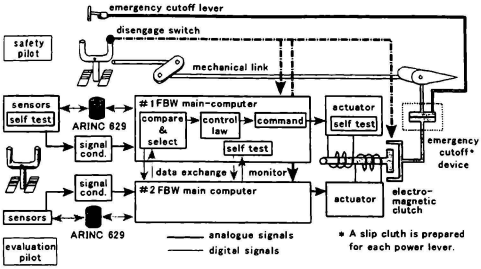


Fig.8 Duplex system concept in FBW system

Actuators

Two types of electrical actuators were developed for the FBW system. One is for the control surfaces and the other is for the engine power levers. Both types of actuators build in a motor with two sets of coils. To cut down the cost for development and maintenance, the same actuator is installed for each control surface with a different gear ratio. The shape was fit to the space for the aileron actuator at the root of main wing. The results of functional tests are shown in fig.7.

Data bus

MuPAL-α adopted the ARINC 629 data bus. The maximum data communication rate of 2Mbit/s is the double the MIL-STD-1553B data bus. In addition, it is very easy for the ARINC 629 to add or remove a terminal, which is a big advantage for an experimental aircraft.

Safety aspect

Primary sensors, a signal conditioner, a data bus, the FBW main-computer and actuators were designed as duplex systems (fig.8). To reduce the probability of shutdown, each component has self test function as much as possible. Two sets of FBW main-computers exchange the data and the self test results each other. Usually the data from #1 sensors are used for calculation in both channels. In case a failure is found in one of #1 sensors, the data from #2 sensors are used alternatively. Two sets of coils in each actuator are commanded separately by the corresponding FBW main-computers. The output from two coils are summed and transmitted to a control surface or an engine power lever via an electromagnetic clutch.

The safety pilot can manually disengage the FBW system whenever necessary and take over the control by using the original mechanical links. The disengage signal lets the electromagnetic clutch out directly. At the same time, the FBW main-computers receive the disengage signal and power off the actuators. The back drive force for each actuator is designed so small that the safety pilot can easily override it even if an electromagnetic clutch malfunctions. Moreover an emergency cutoff device or a slip clutch is prepared.

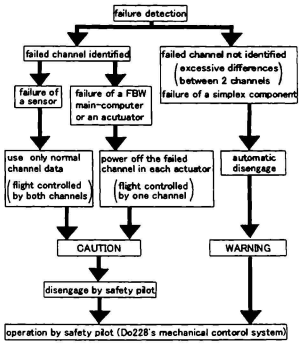


Fig.9 Failure management of FBW system

Fig.9 shows the flowchart for failure management. When a channel in each actuator is powered off, the maximum power of actuators is reduced to 50%. Even if the FBW main-computer fails in detecting a failure, the force-fight between two motor coils inhibits a rapid travel of actuator. The FBW system will be disengaged automatically also when an operational limit for FBW system is exceeded.

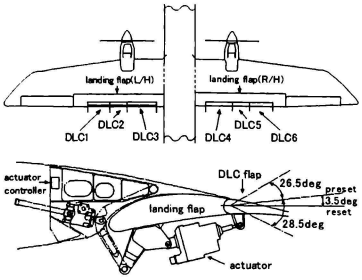


Fig.10 DLC system

DLC System

For the DLC system of MuPAL-α, Dornier examined the following three methods,

- to actuate whole landing flaps,
- to actuate the trailing edges of landing flaps as DLC flaps (flap-in-flap),
- to actuate the ailerons symmetrically,

and recommended the flap-in-flap because

- influences on maneuverability of original aircraft are small,
- required output force for actuators is minimum,
- complexity of actuation system is feasible,
- influences of malfunction of one element are reduced by dividing the DLC surface.

The feasibility study revealed that the operational envelope for DLC system had to be limited due to the strength of original airframe structure (see fig.2 for the clean configuration). When the landing flap is set in the "F1" (5deg down) configuration, the maximum speed is limited under 135kt (69m/s). It is prohibited to operate the DLC system in other flap configurations.

Hardware

The DLC flaps occupy 30% chord and 80% span of the original landing flaps and have 3 segments for each wing (fig.10). All DLC flaps deflect 55deg around the preset position (3.5deg trailing edge up). Each DLC flap is independently controlled by an electrical actuator and its controller according to the same command from a DLC computer. When the DLC flaps are not in use, they are fixed in the reset position, which shapes the original airfoil of landing flaps. The DLC system has no influence on the mechanism of original landing flaps. A DLC control switch in the cockpit is used to preset and reset the DLC flaps before and after experiments.

Performance

The DLC system was designed to generate the lift change amounting to a vertical acceleration greater than 0.2g at 100kt (51m/s). This performance was confirmed through the wind tunnel tests and the flight tests by NAL (fig.21 and fig.22). The actuator was designed to have the output force larger than 2400N (fig.11) and the frequency response bandwidth wider than 2Hz, which keeps the frequency response of aircraft's vertical acceleration flat up to 1 Hz.

Safety aspect

The DLC computer and the position sensor installed in each actuator have two channels. One channel is the control channel to generate all commands for DLC operation, which is connected to #1 FBW main-computer (fig.12). The other is the monitor channel for failure detection, which is connected to #2 FBW main-computer. The DLC computer monitors the difference between two channels about the DLC position commands from two FBW main-computers and the output from position sensors. It also monitors any differences among the positions of six DLC flaps.

If any difference exceeds the threshold, the DLC computer immediately stops all DLC flaps at the latest position and informs the FBW main-computer. DLC

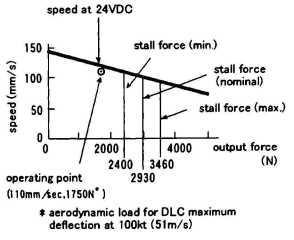


Fig.11 Performance of DLC actuator

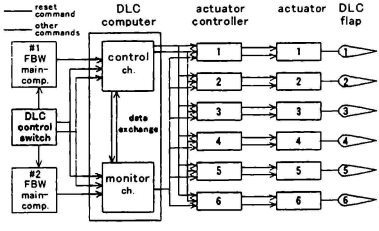


Fig.12 Redundancy of DLC system

flaps are stopped also when the DLC computer is notified of a failure in FBW system. The FBW system is disengaged automatically and then the safety pilot is requested to reset the DLC flaps. As both channels in the DLC computer can generate the reset command, either channel can accomplish the DLC flap reset procedure regardless of FBW system status. MuPAL-α can continue to fly and land safely even if one DLC flap is stuck at the maximally deflected position. The probability for this event was designed to be less than 10^{-5}. The probability for the hazardous or catastrophic events was designed to be less than 10^{-7}.

Second Cockpit

The second cockpit in the cabin was so designed as to provide a flexible research environment for various subjects on human factors. It will be useful also for a simulated landing in high altitude.

The second cockpit has a side stick with force sensors, a rudder pedal and a power lever, whose commands are directly transmitted to the FBW system. The visual images are generated by a graphic workstation and presented to the pilot through a HMD or a 20-inch flat LCD. For each display, fig.13 shows the block diagram and an operation image.

HMD mode

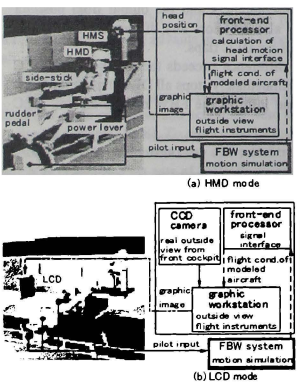
The outstanding feature of HMD is a wide field of view, which can be reoriented according to the motion of pilot's head. The HMD is the only way to create such a visual environment in a small cabin. The HMD can realize various cockpit arrangements virtually by using the computer graphics. The HMD for MuPAL-α has a wider field of view (horizontal: 120deg, vertical: 60deg) and less electric power consumption compared with other products.(fig.14)

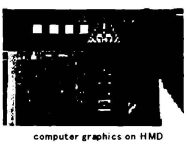
An infrared HMS (Head Motion Sensor) system was newly developed for MuPAL-α. Compared with a magnetic sensor, the output of infrared sensor is more stable and invulnerable to electromagnetic noises. This feature is a big advantage for MuPAL-α, which has to accept various kinds of experimental equipment and rearrangement of them. As the targets of HMS, eight LEDs are so attached on the HMD as to assure a wide effective range for HMS. But through the flight tests, it was revealed that the resolution and the accuracy of current HMS was not enough to make a smooth change of visual image according to quick motion. The research to solve this problem is under way.

From the positions of LEDs, a front-end processor calculates the position and direction of pilot's head and sends them to a graphic workstation. In addition, the front-end processor works as an interface between the FBW main-computer and the graphic workstation. A graphic workstation generates a virtual cockpit with instruments and an outside view. The design of visual images is open to researchers.

LCD mode

In the LCD mode, the front-end processor performs only the interface function. The graphic workstation generates a set of cockpit instruments or an HUD (Head-Up Display) image. Moreover the real outside view, captured by a CCD camera mounted in the front cockpit, can be displayed on the LCD with computer generated symbols (fig.15).




computer graphics on HMD

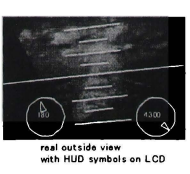

real outside view with HUD symbols on LCD

Fig.13 Second cockpit and its operation image


Fig.14 Head Mounted Display


Fig.15 Flight test scene in the LCD mode

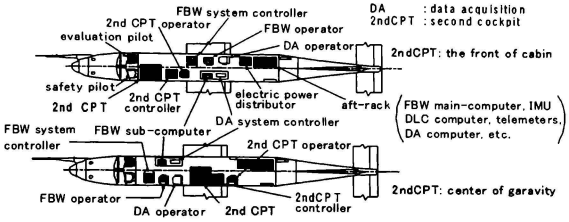


Fig16 Cabin arrangement

Time delay of visual image

Minimization of time delay of visual images is an important subject for the second cockpit. The design requirement was that the time delay between the displayed images and the motion of aircraft or pilot's head should be less than 100ms. It was assessed from a general requirement of ground simulators for middle-sized aircraft. As the in-flight simulator provides more prompt motion cues than ground simulators, it was supposed that the delay of 100ms might cause a discrepancy between the motion cues and the visual cues.

For this design requirement, the delay of 140ms still remains in the HMD mode. However, during the flight tests in the development phase, the delay of 120ms in the LCD mode didn't seem to affect the pilot control at least in the 228 mode (a direct link mode). Further studies including the HMD mode will be carried out.

Cabin arrangement

Fig.16 shows the cabin arrangement of MuPAL-α. The second cockpit can be installed in two different positions in the cabin. One is the front of the cabin. The other is about the center of gravity of aircraft. The front of the cabin will be selected when the motion cues similar to Do228 is preferred. The center of gravity will be selected when the effects of the acceleration caused by aircraft rotation should be eliminated. When the second cockpit is not necessary, it can be removed easily.

Data Acquisition System

Fig.17 shows the block diagram of data acquisition system. The data acquisition computer was newly developed as a VXI bus computer because of the availability of ready-made interface boards for various types of signals. All the measurements, calculation results and system status, which are commonly required for flight data analysis, are on the ARINC629 data bus. So the data acquisition computer receives them as ARINC629 data. To meet the demands for data acquisition from a variety of sensors and on-board systems, the data acquisition computer is equipped with many kinds of interface boards. The time in the data acquisition computer is synchronized with the 1PPS (pulse per second) signal received by the GPS embedded in the IMU.

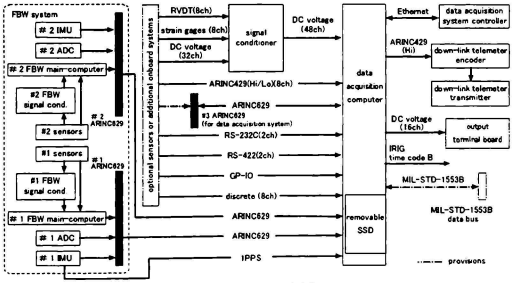


Fig.17 Data acquisition system

The data acquisition computer stamps a time tag on each data and stores it in a removable SSD (Solid-State Disk). The advantage of SSD is its endurance against severe environmental conditions during the flight tests, including the continuous acceleration and the pressure changes. The maximum data acquisition rate is 600Kbit/s. The capacity of SSD is 850MB, which can store all the data from both channels of FBW system for 2 hours. As the data from one channel of FBW system are enough for usual data analysis, the rest can be used for additional data acquisition.

The data acquisition system controller is the same portable workstation as the FBW sub-computer. It is used for the real time data monitoring as well as the data selection for monitoring, recording and telemetering. As the system controller translates the raw data into the physical values and plots them, the operator can change the monitor items and formats freely without any affection on data acquisition.

For the real time monitoring on the ground, a down-link telemeter system transmits 95 items of data at the rate of 25Hz. The data format was adapted to the existing on-ground flight monitoring system.

GROUND SUPPORT SYSTEM

An adequate ground support system is essential to carry forward the flight tests safely and smoothly. For MuPAL-α, the emulation system and the data analysis system was newly developed. As for the flight monitoring system, the existing system is applicable to MuPAL-α.

Emulation System

The objective of emulation system is the pre-flight check of flight control laws and on-board systems in a pilot-in-the-loop condition. According to the actual movement of control surfaces and engine power levers by the actuators, the emulation computer performs the real time calculation of the aircraft motion and the sensor output tainted by each sensor's characteristics, such as delay, noises and errors (fig.18). The simulated sensor output is transmitted to the FBW-main computer and used for the calculation of flight control laws and the indication of flight conditions on the MFDs. Simultaneously the emulation computer generates the outside views based on the simulated aircraft motion. A projection system with three flat screens presents the outside views to the pilot in the cockpit (fig.19).

Data Analysis System

Fig.20 shows the block diagram of data analysis system. A new data processing system is connected to the existing DHS (Data handling System), which includes NAL's standard software for data handling.

After a flight test, the data stored in the SSD are transferred to a workstation for data processing. The

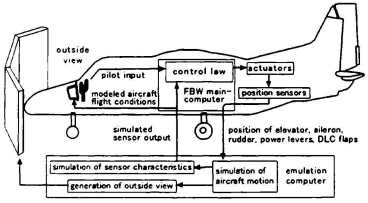


Fig.18 Emulation system

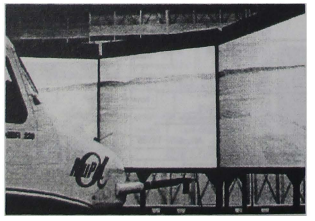


Fig.19 Flat screens for outside views

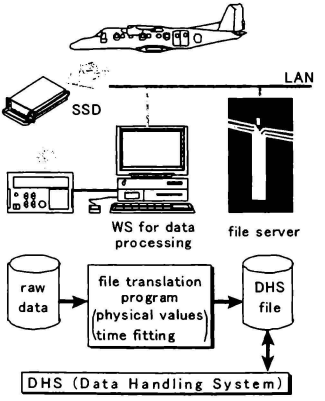


Fig.20 Data analysis system

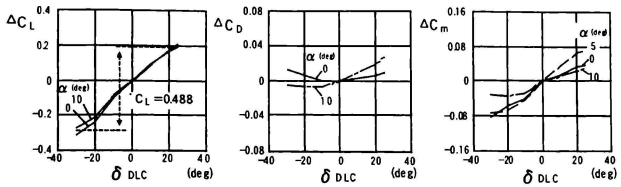


Fig.21 Wind tunnel test results for DLC flaps (Clean configuration)

file translation program translates the raw data into a DHS-formatted file. The DHS files are stored in a file server system. The data necessary for each researcher can be searched, displayed and downloaded by using the DHS.

TOPICS IN THE DEVELOPMENT PHASE

During the development phase, various kinds of tests were carried out to demonstrate the proper functions, performances and airworthiness of MuPAL-α. Some topics in the development phase are mentioned in the following.

Wind tunnel test of DLC flaps

In order to compare the aerodynamic characteristics estimated from flight test data and wind tunnel test data, NAL developed a 1/6-scaled wind tunnel model of Do228-200 with DLC flaps and powered propellers. The wind tunnel test to confirm the static aerodynamic characteristics of DLC flaps was performed in the NAL's low speed wind tunnel, whose test section is 6.5m high and 5.5m wide. At the wind speed of 35m/s, the six elements of forces and moments were measured in various combinations of DLC flap deflection, angle of attack and side slip angle. The pitch angle and the rotation speed of propellers were set so that the thrust generated by propellers was balanced with the drag acted on whole aircraft when both the angle of attack and the side slip angle were zero.

Fig.21 shows a part of wind tunnel test results[6]. In the clean configuration, the change of lift coefficient corresponding to the full deflection of DLC flaps (55deg) amounts to 0.488, which is about 45% of the lift coefficient of total aircraft at 100kt (51m/s). This means that the DLC flaps can generate the vertical acceleration greater than ± 0.2g at 100kt (51m/s), which satisfies the design requirements. The flight test data in fig.22 also demonstrate that the lift change capability of DLC flaps meets the design requirement.

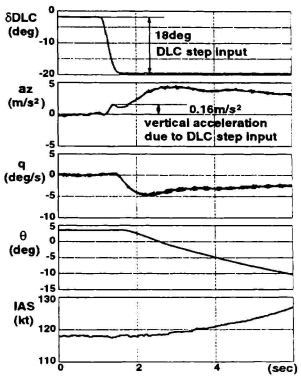


Fig.22 Flight test results for DLC step input

As a part of the proof of airworthiness in the case of malfunction of DLC system, the effects of asymmetric deflection of DLC flaps were measured. The wind tunnel test results proved that the rolling moment caused by asymmetric defection of most outboard DLC flaps could be easily cancelled by the aileron trim function of original Do228. Finally the real landing test with #1 DLC flap fixed at the full up position proved the safety of DLC system.

Flight test of automatic disengage function

To ensure the flight safety, the operational limits for FBW and DLC systems were prescribed so that the operational envelope of original Do228-200 might not be exceeded for any flight control laws. The operational limits consist of the limits for airspeed, bank angle, pitch attitude and vertical acceleration. The limit for bank angle is prescribed as a function of

roll rate and airspeed. Both the FBW and the DLC systems are automatically disengaged when any operational limit is exceeded. A series of ground simulation tests and flight tests proved the validity of operational limits and the safety of recovery operation after the automatic disengagement.

Fig.23 shows the flight test result for the automatic disengagement due to the limit for bank angle. The maximum bank angle was less than 60deg, which is the operational limit of original Do228-200.

FUTURE PLAN

As the first step to attain a sophisticated in-flight simulator, the accuracy of sensors, the aerodynamic characteristics of DLC flaps and the motion control capabilities by FBW and DLC systems will be examined through the filght tests. Then, based on these data, the flight control laws will be refined to improve the accuracy of motion simulation. On the other hand, aiming at the real landing experiments using the FBW and the DLC systems, the landing procedure and its safety will be evaluated by flight tests.

As for the second cockpit, the improvement on the resolution and the accuracy of HMS will be studied, including a try to the HMS with gyro sensors. And a data-link experiment system is being developed for the research on aircraft guidance and operation rules under a future air traffic control and management system.

These plans are expected to be accomplished in a few years. On the other hand, the flight tests requested by the researchers in NAL and other organizations have already started by using the practicable functions. As the first mission for MuPAL-α, the flight demonstration of the ILS approach to a floating structure on the sea is scheduled in June and July in 2000.

CONCLUSION

NAL developed a new in-flight simulator MuPAL-α as an experimental facility for the research and development of advanced aeronautical technologies. The research activities with MuPAL-α have started in April of 2000. The collaboration with other research institutes, universities, aircraft manufacturers and airlines is also expected to be expanded.

We appreciate the cooperation by Kawasaki Heavy Industries, Fairchild Dornier and all people involved in the development of MuPAL-α.

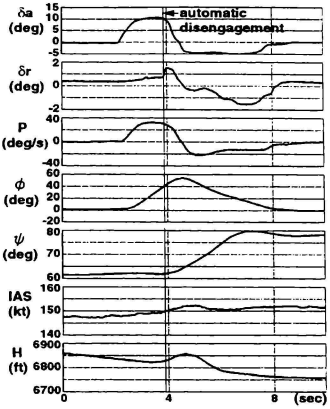


Fig.23 Flight test results of FBW automatic disengage function

REFERENCES

1) Komoda, M., Kawahata, N., Tsukano,Y. and Ono, T., "VSRA In-Flight Simulator -Its Evaluation and Applications", AIAA Paper 88-4605, Sep. 1988.
2) Komoda, M., Kawahata, N., Tsukano,Y. and Ono, T., "Proposal and Flight Evaluation of a New Pitch-Mode Decoupling System", J. of Guidance, Control, and Dynamics, Vol.13, No.6, pp.961-968, 1990.
3) Tsujii, T., Murata, M., Harigae, M., Ono, T. and Inagaki, T., "Development of Kinematic GPS Software, KINGS, and Flight Test Evaluation", NAL TR-1357T, Oct. 1998.
4) Kawahata, N., "Model-Following System with Assignable Error Dynamics and Its Application to Aircraft", J. of Guidance and Control, Vol.3, No.6, pp.508-516, 1980.
5) Funabiki, K. and Sasa, S., "A Head Motion Tracking System using Infrared Sensor", Proceedings of International Conference of Virtual Systems and Multi-Media '96 in Gifu, Nov. 1996.
6) Hozumi, K. and Shirai, M., "Low-Speed Wind Tunnel Testing of DLC-Flaps for MuPAL", Proceedings of 36th Aircraft Symposium, The Japan Society for Aeronautical and Space Sciences, Yokosuka, Japan, 1998, pp.317-320. (in Japanese)

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37289

AIAA-2000-4575

ACT/FHS – AN AIRBORNE ROTORCRAFT SIMULATOR FOR TECHNOLOGY DEVELOPMENT AND RESEARCH

Carl Ockier
DLR, Institute of Flight Research
Braunschweig, Germany

Ulrich Butter
Eurocopter Deutschland GmbH
Munich, Germany

Abstract

The German Aerospace Center (DLR), Eurocopter Deutschland and Liebherr Aerospace Lindenberg are currently engaged in a major research and technology program aimed at the development of a new airborne rotorcraft simulator. The development of this advanced helicopter, called the Active Control Technology Demonstrator and Flying Helicopter Simulator (ACT/FHS) is funded by the German Ministry of Defense, the DLR and industry (Eurocopter and Liebherr). The objective of the ACT/FHS program is to develop an advanced test vehicle for the development of future control technologies and the research into flying qualities and human factors. ACT/FHS is scheduled to perform its first flight in the summer of 2000 and become operational in 2001.

The ACT/FHS airborne simulator is based upon the Eurocopter EC135 helicopter. The EC135 is a light twin-engine helicopter with a bearingless main rotor and a fenestron tail rotor. For the airborne simulation role, the EC135's flight control system is removed and replaced by a fly-by-light control system. The cockpit is modified to accommodate the safety and evaluation pilots and a flight test engineer. A full suite of instrumentation is installed to support the in-flight simulation role.

Introduction

The German Aerospace Center (DLR), Eurocopter Deutschland, and Liebherr Aerospace Lindenberg are developing the Active Control Technology Demonstrator and Flying Helicopter Simulator (ACT/FHS). This helicopter is being developed as an airborne simulator and research platform for the development, demonstration, and evaluation of new flight control and cockpit technologies. ACT/FHS is scheduled to remain in operation for at least 20 years. The ACT/FHS development program is funded by the German Ministry of Defence (MOD), the DLR, and the industry partners Eurocopter and Liebherr.

Presented at the AIAA Modeling and Simulation Technologies Conference, Denver, Colorado, USA, 14-17 August 2000.

Copyright © 2000 by Deutsches Zentrum für Luft- und Raumfahrt e.V. and Eurocopter Deutschland GmbH. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission

The development of ACT/FHS was started in 1996 as a successor to the DLR's BO105 based ATTHeS airborne simulator[1] and the German MOD's BK117 based AVT cockpit technology demonstrator. ACT/FHS will be used to support national and international research and technology programs of the German MOD, the DLR and industry. ACT/FHS will also be used within the framework of German-French cooperation in industry and research, where it is intended to replace the French Dauphin 6001 airborne simulator.

The ACT/FHS helicopter is based on the Eurocopter EC135 helicopter. The transformation of a basic EC135 helicopter into the ACT/FHS research platform requires significant modifications. The mechanical control system is removed and replaced with a full authority fly-by-light system with mechanical backup. The cockpit is modified to accommodate a safety pilot, an evaluation pilot, and a flight test engineer. The aircraft is fully instrumented and equipped with an experimental flight control computer, data acquisition systems, and experimental displays. In addition to the aircraft modification, a complete ground support system is created.

ACT/FHS is expected to make its first flight in the summer of 2000, with a first application phase starting in the first half of 2001 and full system delivery in the second half of 2001.

User requirements

The use of ACT/FHS will concentrate on the investigation and technical assessment of the technological feasibility and operational benefit of essential technologies for future helicopter systems[2]. ACT/FHS is designed to cover a wide range of applications, so that it can be used in all phases of development at research establishments, industry, and government test centers.

The spectrum of user applications covers three elementary stages in the development of a new system:

- *Airborne simulation*: Airborne simulation is an excellent tool for performing basic and applied research in handling qualities, controls, displays, and human factors. To permit making the rapid changes required in

the research environment, a high degree of flexibility must be provided for the airborne simulation role.

- *System development and integration*: The development and integration of new active control technology components, new flight control laws, and new cockpit systems is a second key area of ACT/FHS usage.
- *Technology demonstration*: Technology demonstration represents the final step in using ACT/FHS for the introduction of new technologies. It encompasses demonstrating the functionality and operational benefit of new technologies up to the point of certification.

Each of these three application areas places different requirements on the ACT/FHS platform. Fig. 1 shows the criticality levels and system behavior expected during each of the ACT/FHS applications.

Application	System Standard	Criticality Level	Failure Behavior
Airborne Simulation	Experimental	Non-Essential	Fail safe
System Development	Development	Essential	Fail-operate Fail safe
Technology Demonstration	Operational	Critical	Fail-operate Fail safe

Fig. 1: User requirements and safety standards

System concept

The ACT/FHS user requirements call for a digital flight control system that provides full flexibility with respect to configuration changes and hardware/software modifications for the airborne simulation role and at the same time provides the operational levels of safety needed for the technology demonstration role. These requirements are satisfied by a hierarchical system architecture[3], where a "core system" provides operational levels of safety and an "experimental system" provides the required flexibility. The general system architecture is presented in Fig. 2.

The core system must meet the stringent civil certification requirements, which demand that the likelihood of a catastrophic failure (i.e. loss of control) is less than once per 10^9 flight hours. This could be achieved with a dissimilar quadrupled fly-by-light (FBL) system. To further allow for uncertainties in the applied technology, the FBL system is supplemented by a mechanical backup control system.

To provide the required flexibility, it had to be possible to use a simplex experimental system with a criticality level of no more than minor. This involved the implementation

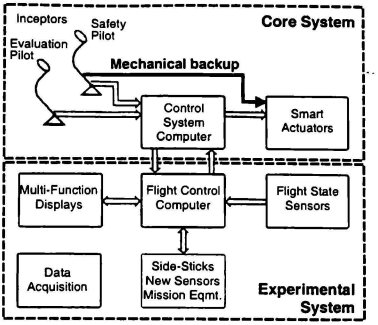


Fig. 2: Basic system architecture

of a number of safety features in the core system, the use of flight envelope restrictions (height, actuator rate, etc.), and continuous monitoring of the control inputs by a hands-on safety pilot. If more relaxed flight restrictions are required or operational levels of safety must be demonstrated, the experimental system can be operated with other levels of criticality.

Fig. 3 shows the subsystem criticality levels and associated limitations. Although the FBL core system is designed to meet a failure probability of 10^{-9}, the actual failure probability in combination with the mechanical backup is reduced to 10^{-7} (due to a 10^{-8} probability of involuntary engagement of the mechanical backup mode).

System	Criticality		Limitations
Experimental System	Minor (current realisation)		Envelope limits SP Hands-on
	Major ($<10^{-5}$) (future redundant realisation)		SP Hands-on
	Hazardous ($<10^{-7}$) (future redundant realisation)		SP Alert
Core System	Hazardous ($<10^{-7}$)	Catastrophic ($<10^{-9}$)	None
Mechanical Backup	Hazardous ($<10^{-7}$)		

Fig. 3: Criticality levels for the different subsystems and the associated limitations.

The EC135 helicopter

The basic airframe for ACT/FHS is an Eurocopter EC135 helicopter. The EC135 is a modern, light helicopter with a bearingless main rotor and fenestron tail rotor (Fig. 4). ACT/FHS is powered by two FADEC controlled Turbomeca Arrius 2B1 engines. The fully equipped empty



Fig. 4: The EC135 helicopter.

weight of ACT/FHS is about 2000 kg, with a maximum take-off weight of 2835 kg. This allows a crew of three and a full fuel load to be taken with a typical operational endurance of almost 3 hours. The EC135's main rotor system, with an equivalent hinge offset of 8.7%, provides sufficient control power and agility for airborne simulation of future helicopters. Roll bandwidth is approximately 3.7 rad/sec; pitch bandwidth is approximately 1.8 rad/sec[4].

Since the flight profile of ACT/FHS is expected to be atypical, a flight usage monitoring system is used to monitor fatigue damage. Data from strain gauges on gearbox, rotor mast, control linkages and tail boom and from torque gauges on the engines and fenestron shaft are continuously recorded by an automatic data collection system.

The ACT/FHS cockpit

For ACT/FHS, the EC135's standard cabin for two pilots and five passengers is transformed to a three-man cockpit. The safety pilot (SP) occupies the left hand pilot's station, the evaluation pilot (EP) occupies the right hand pilot's station, and a flight test engineer's (FTE) station is located in the center of the cabin, behind the pilots (Fig. 5). Experimental equipment is located behind the flight test engineer's station. All crew have crashworthy seating and a radio control panel. The *Avionique Nouvelle* glass cockpit[5] was selected as the standard cockpit instrumentation for ACT/FHS. This cockpit layout, which consists of fully integrated PFD and NAV displays, a warning panel, a systems panel, and a single pointer engine limit indicator, was chosen to alleviate the safety pilot's workload while maximizing his outside field of view. The primary flight instruments (PFD and NAV display) are mounted in front of the safety pilot; engine and system panels, master warning and back-up instrumentation are in the center console. An additional core system control panel is mounted in the center console. The entire cockpit is designed to be compatible with the use of night vision goggles.

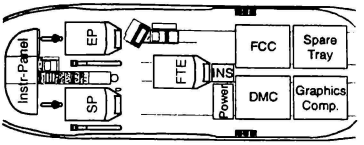


Fig. 5: Top view of the ACT/FHS cockpit and cabin layout (gray areas show ACT/FHS specific equipment)

The evaluation pilot's station

The evaluation pilot's station is modified to have a single 10" flat panel display and a multifunction control and display unit (CDU). The evaluation pilot's display can be freely programmed to show standard instrumentation, quicklook data, sensor images, or advanced display layouts. The CDU is used to interface with the experimental system. Additional test switches are installed on the flight control grips.

The flight test engineer's station

The flight test engineer's station is located behind the pilots' stations, where the FTE has a good view of the instrumentation panel. To his right, the FTE has a console which houses a 10" flat panel display, a multifunction control and display unit, and a radio control unit. These are identical, independent units, with the same functions as the EP's display and CDU. If required, the FTE station can be transformed to a third pilot station by installing side arm controllers and pedals.

Core flight control system

The fly-by-light core system consists of the COre System computer (COS), the smart actuators with Actuator Control Electronics (ACE), the primary flight controls and LVDT position sensors, the evaluation pilot's trim system, and a Core system Control Unit (CCU)[6]. Depending on the selected mode, the control positions of either the safety pilot, the evaluation pilot, or the flight control computer are converted into actuator position commands by the COS and transmitted, via optical data link, to the smart actuators for each of the axes. Additional Hydraulic Commutation Units (HCU) enable transition into the mechanical backup mode or selection of the mechanical control backdrive.

Fig. 6 shows the architecture and operation of the core system. The control signals of safety and evaluation pilot are measured via LVDTs by the COS. From the stick positions, and based upon the mode selected on the core system control unit, the actuator position command is calculated and sent to the ACEs through optical fibers. The ACE then controls the hydraulic actuator to the commanded position.

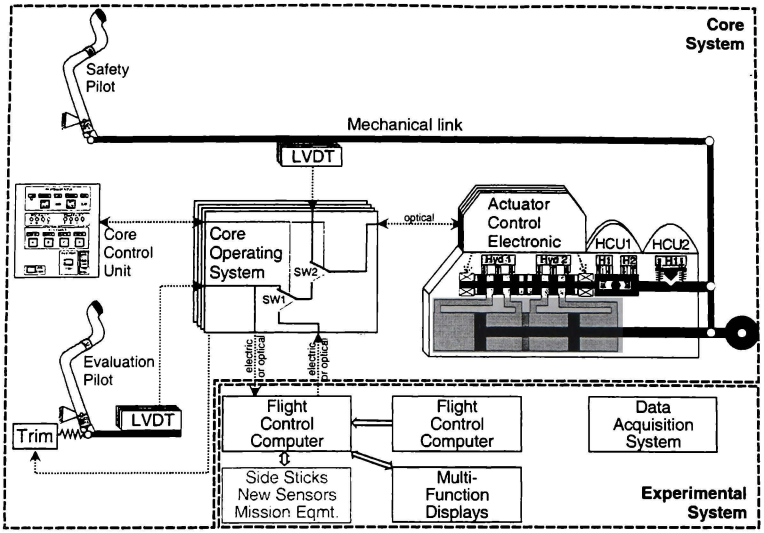


Fig. 6: Simplified diagram of core system architecture (system shown in Safety Pilot Mode, one axis only)

Fig. 7 shows the operating modes of the ACT/FHS core system. Three main operating modes are distinguished (shows as large circles in Fig. 7):

- *The safety pilot (direct) mode*: In this mode, the safety pilot controls the aircraft through the FBL system (as shown in Fig. 6). Although control inputs by the safety pilot move the mechanical control system, the link between stick and actuator servo is open in this case. This is achieved by opening both HCUs 1 and 2.
- *The evaluation pilot (direct) mode*: In this mode, the evaluation pilot controls the aircraft directly through the FBL system. At the same time, the safety pilot's controls are backdriven by the actuator output. This is achieved by activating HCU2 (in Fig. 6, SW2 and HCU2 would be down, SW1 would be up).
- *The experimental mode*: In this mode, the COS obtains the control command signals from the FCC, which may compute these from the evaluation pilot's control positions. As in the evaluation pilot's direct mode, activation of HCU2 causes the safety pilot's controls to be backdriven (in Fig. 6, SW1, SW2, HCU2 would be down).

A fourth mode is the mechanical backup mode, which will be discussed separately.

Transitions between the modes are controlled by the CCU and switches on the safety and evaluation pilot's controls. To select a "higher mode" (from safety pilot mode to evaluation pilot mode or experimental mode and from evaluation pilot mode to experimental mode), the desired mode must pre-selected on the CCU (Fig. 12). After pre-selection, the commands from the pre-selected mode are synchronized with the current actuator position (i.e. the

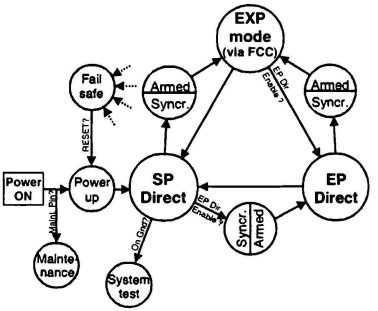


Fig. 7: ACT/FHS core system modes

pre-selected command signal is made approximately equal to the current actuator signal equal to prevent large transients during the mode switch). For the transition from the safety pilot to the evaluation pilot's direct mode, synchronization consists of bringing the SP's and EP's controls in the same position, by driving the EP's trim motors. For mode transitions into the experimental mode, the EP's trim motors or software synchronization can be used. During synchronization, the pre-selection lights on the CCU flash to indicate that synchronization is in progress. Once the current and future modes are synchronized, the lights on the CCU continuously illuminate to indicate the mode is armed and a mode transition can take place. The pilot in control then initiates the actual switchover by pressing a button on his collective lever.

To select a "lower mode", the safety or evaluation pilot can press a switch on the collective lever (or a backup switch on the safety pilot's cyclic). In addition, the safety pilot can take control by trying to override the control forces. This causes force switches to activate, so that the COS transfers to the safety pilot mode. Transfers into lower modes are virtually immediate, to avoid dangerous delays in the case of an experimental system runaway. Transfers into higher modes are passed through a fading function to avoid transients.

Other operating modes are built in system modes, which are not typically used during flight. These include:

- *Power up mode*. After electrical power is applied, a built-in test is run, after which the core system automatically switches into safety pilot mode.
- *System test mode*: A comprehensive system test can be initiated by pressing a button on the CCU.
- *Fail safe mode*: If continuous monitoring detects a failure in a core system lane, the lane will go into fail safe mode. Single lanes can be reset in flight.
- *Maintenance mode*: In the maintenance mode, a memory area where all failures are stored can be read and new software can be loaded

Core system computer (COS)

The core system computer serves as the central element of the FBL control system. The COS controls mode changes, generates the command signals for the smart actuators, and performs most safety functions.

The COS consists of four functionally identical lanes, each serving four ACE lanes (one on each actuator). The COS hardware is housed in two segregated boxes (Fig. 8). Each box contains two dissimilar lanes, with one lane based on the Motorola 68332 microcontroller and the other hardware lane based on the Texas Instruments SMQ320C32 signal processor. All lanes run asynchronous, with a cycle time of 2 ms. Each lane has a number of continuous tests and watchdog timers installed. An

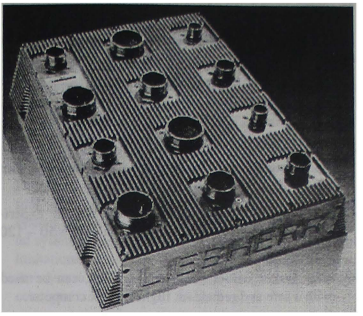


Fig. 8: COS box housing two COS lanes (Photo Liebherr Aerospace Lindenberg)

optical cross-communication between the lanes exchanges mode switching information.

shows a diagram of the signal flow within the COS. The signal path starts at the safety and evaluation pilot's LVDT control position sensors. Within the COS, the LVDT signals are demodulated and A/D converted. The evaluation pilot's control positions are passed to the experimental flight control computer where they may serve as an input to the control law. Depending on the selected mode, the control positions of the safety or evaluation pilot or from the flight control computer pass through a phase shifting filter to avoid pilot induced oscillations in the roll axis. At the end of the control path, a constant 190%/sec rate limiter ensures that actuator command rates do not exceed the hydraulic capacity of the helicopter.

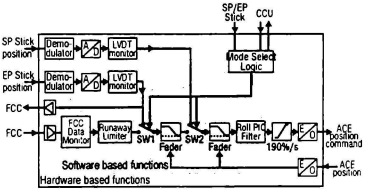


Fig. 9: Principal signal flow within the COS

In the experimental mode, the control commands from the FCC are monitored with respect to validity, parity and update. In addition, the signals are passed through a runaway limiter to prevent faulty FCC signals from structurally damaging the helicopter. The algorithm used by the runaway limiter restricts the actuator rate for large and fast signals, but not for slow or for fast short signals.

as shown in Fig. 10. As such, the runaway limiter provides the largest possible flight envelope without endangering the aircraft. Four sets of limiters are currently defined:

- One fairly restrictive limiter permits experimental mode operations throughout most of the flight envelope.
- One restrictive limiter permits experimental mode operations throughout the speed entire envelope with a 50 –120 ft altitude restriction.
- One minimally restrictive limiter permits experimental mode operations up to 120 kt with a 50 – 120 ft altitude restriction.
- One limiter with no restrictions at all, can be used with a safe and redundant flight control computer.

The current limiter values are determined from simulation, using the assumption that a 400 msec intervention time is needed for the safety pilot to assume control after a runaway. The final limiter values will be fixed after flight testing.

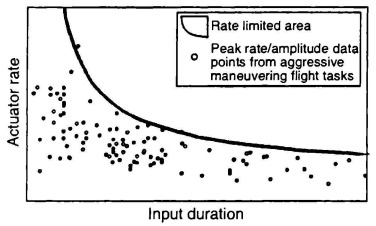


Fig. 10: Operation of the runaway limiter and comparison to actual flight data of comparable helicopters during aggressive maneuvers

Automatic fading functions are attached to the software mode switches SW1 and SW2, to provide a smooth transition between the control command sources considered. Different fading algorithms are depending on the mode and direction of the transition.

Finally, the COS also monitors and controls the evaluation pilot's trim system. In evaluation pilot' direct mode, the EP's trim system is controlled by the COS. In the experimental mode, the COS relays the EP's trim signals from the FCC to the trim motoring unit.

Smart actuators

ACT/FHS has four identical smart actuators: three main rotor actuators mounted on the cabin ceiling (Fig. 11) and one tail rotor actuator in the vertical fin to control the fenestron. The ACT/FHS actuator design consists of a tandem cylinder assembly driven by a quadruplex Direct Drive Valve (DDV) assembly and controlled by the quadruplex Actuator Control Electronics (ACE). By design, the two hydraulic systems are completely segregated. The DDV assembly contains two rotary control valves commanded by a quadruplex electrical rotary torque motor mounted on a single control valve shaft, which controls both valves.

The quadruplex actuator control electronics (ACE) are mounted directly on top of the hydraulic actuator. Each ACE lane receives the applicable actuator command from one COS lane. Each ACE lane drives one coil of the quadruplex torque motor mounted on the shaft of the DDV control valve of the actuator. The actuator is controlled digitally with a two level cascade loop controller, with the outer loop controlling the actuator position and the inner loop controlling the direct drive valve position (which is proportional to the actuator rate). The control valve position commands, as well as the measured control valve position signals, are consolidated across all channels to avoid force fighting on the control valve shaft. By limiting the actuator position error before consolidation, undetected hardware or software failures in a single channel can be compensated by the remaining healthy channels. Control of the outer loop is performed with a cycle time of 2 ms; the inner loop control has a cycling time of 400 µs. Performance of the FBL actuators is comparable to that of the standard EC135 main rotor actuators.

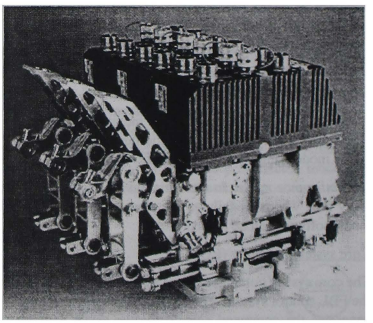


Fig. 11: Main rotor actuators (Photo Liebherr)

Core system control unit (CCU)

The core system control unit (CCU) is the main interface between the pilots and the core system (Fig. 12). The CCU is located in the center console, where it is visible to all crewmembers. The CCU contains switches for the preselection of operating modes, for resetting individual lanes, and for activating test functions. Indicators on the CCU show the active mode, the status of the synchroniza-

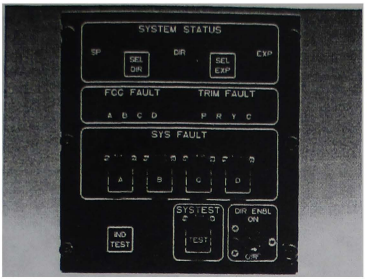


Fig. 12: Core system Control Unit

tion and arming procedure, and system detected faults. In addition, certain status and fault indications are shown on the instrumentation panel and in the central warning panel. Mode transitions are announced by an acoustic signal over intercom

Software development

Core system software is developed following the rules for level "A" functions, according to RTCA/DO-178B and ARP 4754. System requirements are translated into two dissimilar software requirements. Software design and verification are performed by two different teams, with each team designing the software for both hardware variants against one of the two software design documents. This leads to four dissimilar sets of software. All software is written in the C language.

Mechanical backup control system

The mechanical backup control system provides two essential functions in ACT/FHS:

- The mechanical control system serves as a last-resort backup system for the core system. In case of a total core system failure (which should – at least in theory – occur less than once per 10^9 flight hours), the safety pilot can press a switch on his cyclic grip to de-energize the entire FBL system. De-energizing the core system leads to the closure of HCU1 (), which connects the mechanical control rods with the actuator servo input.
- The mechanical control system also provides the safety pilot with feedback of the evaluation pilot's or FCC's control inputs. To achieve this, the mechanical controls are backdriven by the actuators (HCU2 is activated, see Fig. 6).

The mechanical control system of ACT/FHS is completely re-designed from the original EC135 control system. Instead of rods and bellcranks, the EC135 uses flexball cables (a.k.a. Bowden wires). The main rotor flexball cables run from under the cockpit floor via the windscreen center post up to the main rotor actuators in front of the gearbox.

Experimental system

The experimental system consists of the flight control computer, the data acquisition system, the multifunction display system, and the flight test instrumentation. The general system architecture of the experimental system is shown in Fig. 13.

Except for the EC135's basic attitude and air data sensors, the current implementation of the experimental system is simplex. This is consistent with the experimental system being of the lowest criticality level. If the flight limitations imposed by this criticality level are too severe, the system can easily be upgraded using redundancy concepts.

The experimental system is connected to the core system with a set of ARINC 429 100 Kbps connections or with a 2 Mbps optical connection. The main connection between core and experimental system is through the flight control computer, which receives the pilot's control inputs from the COS and generates commands for the actuators. In the current implementation, one FCC listens to all four COS lanes. The ARINC 429 data from the single FCC are split into four wires, one for each COS lane. In addition to the

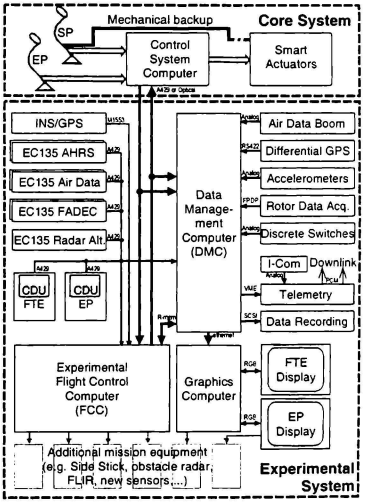


Fig. 13: Experimental system architecture

FCC, the data management computer (DMC) also monitors each of the COS lanes, as well as the FCC output.

The flight control computer (FCC)

The flight control computer (FCC) generates the actuator command positions and trim signals when the core system is in the experimental mode (or when the experimental mode is pre-selected). Typically, the FCC will implement flight control laws provided by the experimenter.

The FCC is a VMEbus system based upon a 200 MHz Power PC processor. The FCC is connected directly to the COS and to the EC135s basic attitude and air data sensors by via ARINC 429, to the inertial navigation unit via MIL 1553, to the data management computer through reflective memory, and to the CDU's. A large number of VMEbus slots are available on the FCC, to allow for other computers and mission equipment to be connected. Typical cycle time of the FCC is 5 ms.

The data management computer (DMC)

The data management computer is the backbone of the data acquisition system. It collects data from all sensors and the FCC, processes and converts the data into engineering units, records the data on disk, sends it to telemetry and the graphics computer.

The DMC is a VMEbus based computer, with two Power PC CPUs and a whole array of interface cards. The DMC is connected to virtually all components of the experimental system. The DMC serves a 2 Gbyte flash disk storage device through a SCSI interface and also generates data for telemetry and for the display system. The data management computer is operated through the control and display units (CDUs), which are directly connected to the DMC. The DMC has a standard cycle time of 2 ms.

The display system

ACT/FHS provides two programmable cockpit displays, one for the evaluation pilot and one for the flight test engineer. The display system can be used as a multifunction display and for the presentation of flight instruments, time history and quicklook data, digital maps, and test specific information (Fig. 14). The displays are driven by a ruggedized Silicon Graphics O2 computer with dual head graphics card. The O2 graphics computer is connected to the data management computer via ethernet.

The displays are rugged active matrix LCD color displays (Barco MPRD 126 HD/NV) with a 26-cm (10") screen diagonal and a resolution of 800x600 pixels (SVGA). Brightness is 200 fL for sunlight operations with a 3000:1 dimming ratio for night operations. The displays are fully NVIS B color compatible. A scaler input allows video signals of almost any format to be mixed with the standard RGB input. 6 function soft keys are available on each side of the display.

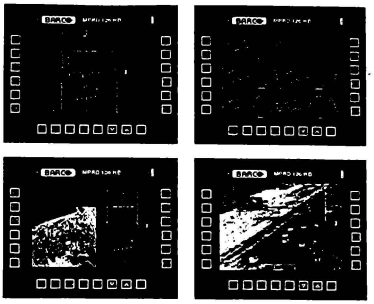


Fig. 14: Four typical uses of the display system

The control and display units (CDU)

The operator's main interface with the experimental system is through one of two control and display units. The CDUs are used to control computer initialization, to set and adjust parameters within the experimental system, and to display faults and status information. The CDUs are connected to the DMC and FCC.

The CDUs are standard Smiths Industries MCDUs used in a large range of aircraft and modified to suit the ACT/FHS key labeling requirements. The NVIS compatible CDUs include a full range of alphanumeric keys, special purpose function keys, call lights, and a high brightness color text display for 14 lines of 25 characters. The CDUs are operated in accordance with the ARINC 739 standard.

Flight test instrumentation

Since ACT/FHS is designed not only as a technology demonstrator, but also as a technology validator and research vehicle, highly accurate flight state data have to be acquired. This is achieved through the installation of the following sensors:

- *Air data boom*: An air data boom with a swiveling pitot-static tube and angle of attack and sideslip vanes (Space Age Control 100510) is mounted on a noseboom extending about 2 m in front of the aircraft. Above 20 kt, the airflow at this location should be sufficiently clear of the airframe and rotor wake interactions. The gimbal-mounted swivel head aligns the Pitot-static tube with the airflow and allows accurate speed measurements for angles of attack or sideslip of up to 30°. Two high precision Setra pressure transducers in the nose of the helicopter transform the pressures into analog voltages. Vanes at the body of the probe measure angle of attack and sideslip. The vanes are connected to potentiometers, which provide an analog voltage as a function of vane angle. To complete the air data measurements, a Rose-

mount air temperature sensor is mounted at the base of the boom.

- *Inertial navigation system (INS)*: Accelerations, angular rates, attitudes, ground speed, and aircraft position are obtained with the Honeywell H764G embedded INS/GPS system. This laser-gyro based strapdown platform was selected because of its high accuracy and its ability to deliver "raw" data at rates of up to 256 Hz. Other features of the INS include the ability to provide pure INS, pure GPS, and combined INS/GPS solutions. Data are transmitted through a fast MILBUS 1553B interface. The GPS antenna is mounted on the cabin ceiling.

- *Differential GPS*: Accurate position data are obtained from a differential GPS sensor (NavSymm Sharpe XR6). This sensor was selected because of its sub-meter accuracy and its 10 Hz output rate. The XR6 obtains its accurate solution from extensive carrier phase tracking. The GPS antennas are mounted on top of the tail fin.

- *Accelerometers*: To obtain the acceleration at the evaluation pilot's station and as a redundant sensor to the INS, a set of micromachined servo accelerometers (Endevco MSA 100) arranged in a triangular mounting are installed behind the evaluation pilot's seat. The accelerometers use a three layer micromachined silicon sensor, which rebalances a reference mass to the zero position between the upper and lower layer. The control force needed for zeroing the mass provides the analog data signal.

- *Rotor data acquisition*: Although extensive use of rotating frame main rotor data is not initially foreseen, a 25 channel rotor data acquisition system, developed by Manner GmbH, is installed. The system consists of three parts: a data acquisition system on the rotor, an inductive transmission device at the rotor shaft, and a receiver and control system. Via inductive transmission, set-up data and power is transferred to the data acquisition system on top of the rotor head. This data acquisition system is able to sample and convert rotor azimuth and 24 analogue sensor signals from strain gauges or potentiometers to digital values, with sample rates up to 2000 Hz. The digital data is transferred by short distance telemetry to the data reception unit in the rear cabin of the helicopter.

Basic aircraft sensors

Several basic aircraft sensors are also available to the experimenter:

- *Air data units (ADUs)*: Two independent air data units (Sextant Avionique ADU 3000) measure static and differential pressure as well as air temperature from a combined pitot/temperature probe on each side of the airframe. From these data, each ADU calculates altitude, altitude rate, mach number, true air speed and computed air speed. The data are transmitted via ARINC 429 at a refresh rate of 20 Hz.

- *Attitude and heading reference systems (AHRSs)*: Two independent AHRS (SFIM APIRS 2101) provide attitude, magnetic heading, angular rates and acceleration data. Data are transmitted on ARINC 429 at a rate of 128 Hz for acceleration and angular rate data and 64 Hz for attitude and heading data.

- *Radar altimeter*: One radar altimeter (Bendix/King KRA 405B) provides height above ground up to 2500 feet. Output is ARINC 429 at 50 Hz.

- *FADEC data*: The engine control computer (FADEC computer) provides engine sensor and control data such as gas temperature, generator speed, free turbine speed, engine torque and fuel metering position. These data are transmitted at 40 Hz in ARINC 429 format.

- *Intercom signal*: The analog signal from the intercom is available to the data management computer and can be recorded in the digital data stream and telemetried to the ground station.

Telemetry

The telemetry system provides the flight test observation station with real time data and intercom from the helicopter. The digital PCM data link is capable of handling up to 3 Mbit/sec. Intercom can be transmitted with the PCM data stream. The PCM encoder can be parameterized through software to allow ACT/FHS to work with different telemetry receivers. The PCM encoder is developed by Hentschel Systems and is housed on a VMEbus board in the DMC. A Teledyne PCM merger allows audio and video signals to be mixed with the standard PCM data stream. To maximize the likelihood of telemetry signal reception, the telemetry signal is sent over two frequencies and two different antennas, one on the tail fin and one on the cabin ceiling or noseboom.

Software

Software for the experimental system is developed for the FCC, the DMC, and the graphics computer. The VMEbus based computers (FCC and DMC) use the VxWorks™ real-time operating System and are programmed in C. The graphics processor uses the Silicon Graphics IRIX™ operating system with programming in C using Open-GL or with the VAPS™ rapid prototyping tool. The software development and quality assurance standards are defined in accordance with RTCA D0-178B. The complete software development process is supported by the use of CASE tools, code checkers, and configuration management tools.

The experimental system software is designed to distinguish between a lower and an upper software layer. Basic support functions like sensor I/O, storage, telemetry, mode switching, etc. reside in the lower lever software. The upper level software contains user specific programs

such as control laws. A carefully designed software interface permits the development of user software, without having to consider lower level details. In addition, the software is strongly parameterized so that the user can modify the software behavior without having to change the source code.

Support equipment

The ACT/FHS system also includes ground based support equipment. The two most prominent elements of this support equipment are the hardware-the-loop simulation facility and a mobile flight test observation unit.

The helicopter simulation facility

The helicopter simulation facility at DLR is designed as a hardware and software in the loop test facility for ACT/FHS[7]. This facility was created:

- to allow software and hardware to be developed and verified before flight in a real-time pilot-in-the-loop environment;
- to provide pre-flight training for aircrew;
- to provide a test preparation facility for technical specialists and flight test engineers.

To achieve this, a fixed base simulator was built around a BO 105 cockpit, which was modified with a set of original ACT/FHS inceptors, an evaluation pilot's display, a CDU, a core system control unit, and simulated Avionique Nouvelle instruments. An active force feel system was installed to replicate the control forces in ACT/FHS and to simulate the mechanical control backdrive mode. A state of the art visual system provides a field of view that extends from 45° to the left to 135° to the right and from 20° up to 45° down (Fig. 15). The simulator is driven by a high fidelity real-time EC135 model, with a comprehensive simulation of the core flight control system. The experimental system hardware is practically identical with that of the airborne system to allow all experimental system software to be pre-flight tested in the simulator.

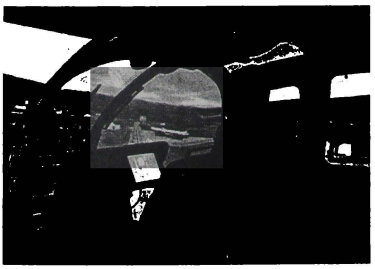


Fig. 15: The ground based simulation facility (during pre-completion testing)

The flight test observation unit

To permit real-time monitoring of flight tests with ACT/FHS, a mobile flight test observation unit is being built. This unit provides facilities for real-time monitoring and off-line data analysis. It consists of two modules: a real-time monitoring station and a telemetry station. The real-time monitoring station provides room for up to 4 engineers with 3 terminals for real-time monitoring of telemetry data, an off-line data conversion and analysis station, communications equipment, and a software development computer. The telemetry station houses a telemetry tracking system, PCM decoder, data storage unit, communications equipment, space for up to two operators, and a small desk area. Each station is housed in a transportable container, to allow operations at remote test sites.

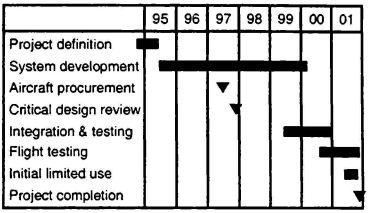


Fig. 16: Time schedule for the development of the ACT/FHS airborne simulator

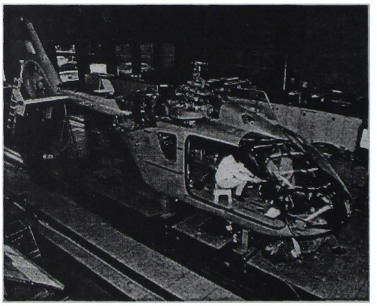


Fig. 17: The ACT/FHS helicopter during system integration at Eurocopter Deutschland

Status and outlook

Fig. 16 shows a time schedule of the ACT/FHS development. By the summer of 2000, development of ACT/FHS is completed, with final assembly, system integration and ground testing being fully under way (***Fig. 17***).The first flight of ACT/FHS is scheduled for the summer of 2000,

with developmental flight testing to be completed by the end of the first quarter 2001. After flight-testing, the final core system software must be developed and verified, a process which is scheduled to require 6 months. During this software development phase, an initial user program aimed at the development of basic model following control laws will be completed. With the final core system software, ACT/FHS will be fully operational in the fall of 2001.

References

[1] Pausder, H.-J., Bouwer, G and von Grünhagen, W., "ATTHeS in-flight simulator for flying qualities research," International symposium "in-flight simulation for the 90's", Braunschweig, Germany, July 1991.

[22] Pausder, H.-J., Butter, U. and Gollnick, V., "ACT Demonstrator / Flying Helicopter Simulator – A new testbed for flight control and cockpit technologies," 52nd AHS Annual Forum, Washington D.C., USA, June 1996.

[3] Butter, U., Pausder, H.-J. and Steinmaier, F., "Safety concept of technology demonstrator ACT/FHS," AHS Annual Forum, Virginia Beach, USA, May 2000.

[4] Kampa, K. Enenkl, B., Polz, G, and Roth, G, "Aeromechanic aspects in the design of the EC135," 23rd European Rotorcraft Forum, Dresden, Germany, September 1997.

[5] Achache, M, Cognet, J., Germanetti, S. and Lanteaume, M., "Avionique Nouvelle – Evaluation and certification of modular avionics applied to Eurocopter light and medium helicopters," 54 AHS Annual Forum, Washington D.C., USA, May 1998.

[6] Steinmaier, F., Bickel, N. and Pausder, H.-J., "Survivable fly-by-light control system, ACT/FHS," RTO Meeting Proceedings 47, AC/323(SCI)TP/22, April 2000.

[7] Hamers, M., Bouwer, G, von Grünhagen, W., "System simulator for the DLR's in-flight simulator ACT/FHS – software and hardware realization," AHS Vertical Lift Aircraft Design Conference, San Francisco, USA, January 2000.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37290

AIAA-2000-4576

IN-FLIGHT SIMULATION AS TOOL FOR THE DESIGN OF FUTURE PILOT ASSISTANCE SYSTEMS*

Peter Hecker
Institute of Flight Guidance, German Aerospace Center, DLR
Lilienthalplatz 7, D-38108 Braunschweig, Germany
Email: peter.hecker@dlr.de

ABSTRACT

DLR is developing pilot assistance systems to increase safety and economy of aviation. All phases "gate to gate" are covered by special assistance modules plugged into a core data management system. The system architecture will be introduced as well as some of the most important assistance functions like "Advanced Flight Management" or "Enhanced Vision". In addition the validation of pilot assistance systems using flight simulators and test aircraft with in-flight simulation capabilities will be presented.

ABBREVIATIONS

ADVISE	Advanced Visual System for Situation Awareness Enhancement
AFCS	Automatic Flight Control System
AFMS	Advanced Flight Management System
AHMI	Airborne Human Machine Interface
ATC	Air Traffic Control
ATM	Air Traffic Management
ATTAS	Advanced Technology Testing Aircraft System
ATMOS	Air Traffic Management and Operations Simulator
CAMA	Crew Assistant for Military Aircraft
CASSY	Cockpit Assistant System
CDU	Control Display Unit
EFMS	Experimental Flight Management System
EVS	Enhanced Vision System
FMS	Flight Management System
IPA	Intelligent Pilot Assistant
ND	Navigation Display
PD/2	Phare Demonstration 2
PFD	Primary Flight Display
PHARE	Programme for Harmonized Air traffic management Research in Eurocontrol
TARMAC	Taxi and Ramp Management and Control
TCAS	Traffic Alert and Collision Avoidance System

INTRODUCTION

Recent prognoses foresee a dramatic increase of air traffic within the next decade. Keeping the safety of aviation at a high, but constant level, this means immediately a severe increase of absolute numbers of accidents. Therefore it easily can be seen, that a tremendous effort has to be made in order to keep aviation the highly accepted and economically attractive transportation system as it is right now.

The analysis of recent statistics has shown, that a major part of accidents was caused by a chain of reasons, where the human operator was involved in some sense. Looking more into the details, it can be seen, that the "computerization" of the cockpit, which began in the late 70s, brought lots of highly specialized, but isolated systems for individual tasks into the cockpit. The heterogeneous design and the number of separated subsystems increased the workload of the aircrew.

Again, upcoming new Air Traffic Management (ATM) concepts, such as free-flight, are bringing new tasks like separation assurance or co-operative flight planning into the cockpit. Traditionally new separate systems would have to be designed and built for these tasks. This approach is however reaching its limits, as the situation awareness of the pilots is hindered by having to collect the required information from a number of devices with inconsistent display formats. Inputs into the system are also becoming more and more difficult, as the individual systems tend to have independently developed and poorly integrated user interfaces.

Therefore DLR's Institute of Flight Guidance is looking deeper into the design and development of so-called "pilot assistance systems" since many years, where an integrated system combines all required assistance functions under a single human machine interface[6].

* Copyright ©2000 by German Aerospace Center DLR. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission

An overall concept including a very specific system architecture was developed in order to provide pilot assistance during all phases of flight from "gate-to-gate". In this architecture the assistant functions are implemented in function modules grouped around a core system consisting of a central data pool and a module manager. For several tasks of flight guidance, a number of highly specialized modules is being developed and tested in national and international research programs. Within these programs various modes of simulation including in-flight simulation were utilized in order to keep an optimal balance between results and costs.

The presented paper will give an overview about the concepts of pilot assistance and some of the most important modules involved. In a second step the design, development and evaluation of two major elements will be shown in more detail. A special emphasis will be given to the validation of these modules using various modes of simulation including in-flight simulation.

PILOT ASSISTANCE GATE-TO-GATE

CONCEPT

In order to fulfill a given mission an aircrew has to solve a large number of usually quite complex tasks of flight guidance. In a simplified scheme as shown in Figure 1 these tasks can be regarded as a guidance loop the pilot has to run through continuously. The major elements of this loop are

- situation assessment,
- plan generation according to the present state and the given mission,
- decision which plan to follow,
- execution of the plan, and finally
- monitoring if the flight is following the selected plan.

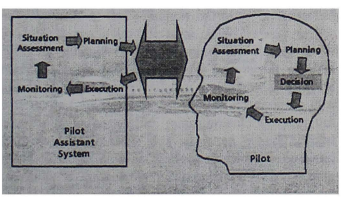


Figure 1: Pilot Assistant

In order to assist aircrews performing this loop, a concept for an "electronic co-pilot", a so-called Intelligent Pilot Assistant (IPA) has been developed. As depicted in Figure 1 this IPA performs exactly the same loop as the pilot except the task of decision making. In commercial aviation and many military applications this step still has to remain within the authority of the aircrew. Nevertheless the concept can provide complete autonomous operation modes, as they might be required for future single piloted cockpits or unmanned aerial vehicles.

The IPA concept enables two significant capabilities:

First, the IPA is able to assess the situation and to generate valid plans autonomously. Especially during critical flight phases with high workloads this capability is most essential. As an example conventional, Flight Management System (FMS)-based flight guidance suffers from the volume and complexity of data to be entered into the management unit. The crew first has to assess the situation by reading several, independently operating systems and by communicating via voice with Air Traffic Control (ATC). Then the pilot has to go through a planning procedure and then has to enter the plan into the FMS by typing sequences of commands and parameters. Performing tasks requiring quick reaction like modifications of the flight plan during approach and landing the crew very often dispenses with FMS and performs the necessary maneuvers manually.

In this case the IPA's situation assessment and planning functions could assist the crew substantially. By means of sensors, databases, internal state data and data brought on-board via datalink the assistance system continuously is able to asses and interpret the situation. Via datalink or via voice control the IPA can get the instruction to change the flight plan. A valid plan can be computed and offered to the pilot automatically. The pilot just has to accept and activate the IPA's proposal or to reject and ask for alternative plans. Figure 2 shows the possible reduction of workload by pilot assistance systems for another example.

The second important feature is the common human machine interface for all functions in the cockpit. A homogenous concept of communication between pilot and assistance systems ensures, that the pilot is able to obtain just the information he needs regarding a specific situation. The way, information is presented to the pilot either by passive displays, interactive displays or acoustic or tactile communication is harmonized and optimized regarding the needs of the human operator.

The benefits of such systems have been demonstrated with the "Cockpit ASistant System" (CASSY) for civil flights under instrument flight rules developed by the University of the German Armed Forces and tested on DLR's Advanced Technology Testing Aircraft System (ATTAS) as well as the successor system "Crew Assistant for Military Aircraft" CAMA[9]. CAMA was a joint development of the University of the German Armed Forces, the

Elektroniksystem und Logistik GmbH (ESG), DaimlerChrysler Aerospace military aircraft division and DLR's Institute of Flight Guidance. This system has received excellent ratings from military test pilots in two simulator campaigns and was recently flown in DLR's ATTAS aircraft.

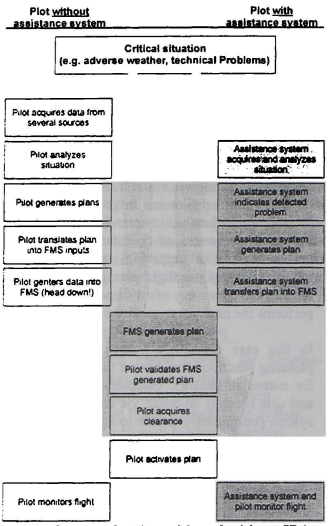


Figure 2: Course of action with and without IPA

ARCHITECTURE

A generic architecture for intelligent pilot assistance has been developed at DLR's Institute of Flight Guidance (see Figure 3) [8,11]. Within this architecture assistant functions are implemented as modules grouped around a core system consisting of a central data pool and a module manager. This data pool contains all relevant data available on board. Therefore every module connected to the pool is able to access any necessary data via the module manager.

Sophisticated data handling methods are implemented in this architecture to allow modules to be notified whenever relevant data have changed. This architecture allows sequences of actions performed by the system to be modeled entirely by specifying which input data are required for certain functions. Changes in these data will then trigger the corresponding functions. This mechanism is used to achieve the situation assessment and automatic response to critical situations.

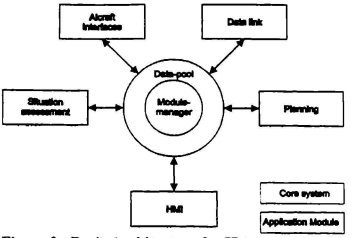


Figure 3: Basic Architecture for IPA

The architecture is very flexible and easily extendible. Because of it's generic character it easily can be used as core for any kind of assistance system. Predefined templates allow a rapid design of new modules and functions. In addition already existing systems like the Traffic Alert and Collision Avoidance System (TCAS) or the Automatic Flight Control System (AFCS) for the domain of flight guidance can be integrated as well.

ASSISTANCE "GATE TO GATE"

Based on this generic architecture DLR's Institute of Flight Guidance is developing a pilot assistance system aiding the pilot during all phases of flight including taxiing. The so-called IPA "Gate to Gate" consists of the already described generic core system containing the data pool and the data management unit. In addition a number of specialized modules for the involved guidance tasks as shown in Figure 4 are under development or have been developed already. These modules cover three domains: interfacing, situation assessment and planing/execution.

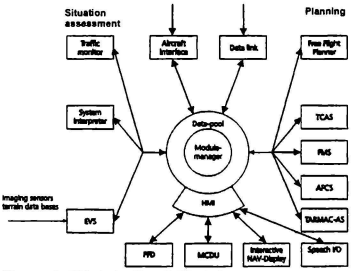


Figure 4: "Gate to Gate" Modules

The interface domain includes all standard interfaces to the typical aircraft systems as well as data links to external parties like other aircraft or air traffic control. In addition various elements of human

machine interfacing are under development like displays, speech recognition and voice output.

The second domain encloses modules for situation assessment. A traffic monitor module observes the surrounding traffic and detects potential conflicts. On board systems are being monitored by a system interpreter module which is able to detect malfunctions and to generate a diagnosis for the afflicted systems. Finally, an weather independent operating Enhanced Vision System (EVS) is observing the airspace and the terrain in front of the aircraft in order to detect the upcoming scenery including unknown obstacles and to provide a navigation update.

Finally, the third domain includes planning modules like a free flight planner and the flight management system. Furthermore assistance modules for the control of aircraft like the "Automatic Flight Control System" AFCS, "Traffic Alert and Collision Avoidance System" TCAS or the "Taxi and Ramp Management and Control – Airborne System" TARMAC-AS are linked to the IPA.

As each of these modules is very complex, two of them will be selected as examples for the development. The following sections will describe the development and the validation of these modules in more detail.

FLIGHT MANAGEMENT MODULES

CONCEPT

The strategic trajectory generation as well as the automatic guidance along this trajectory according to schedule is the domain of the Flight Management System. At the present state of art the FMS suffers from the poor or not existing interfacing with the aircrew and ATC, as described above. A substantial improvement of this situation only can be achieved by introducing a co-operative ATM, where traffic planning modules on the ground are connected to flight planning systems on board the aircraft via data link . In order to investigate the potential and the benefits of these innovative, a Programme for Harmonized Air traffic management Research in Eurocontrol (PHARE) was launched, where several European research organizations studied concepts and technologies supporting this concept of ATM. Within this program an Experimental Flight Management System (EFMS) was developed, where the conventional FMS functionality was extended by co-operative elements. This EFMS was developed further to the Advanced Flight Management System (AFMS)-module used for IPA [2].

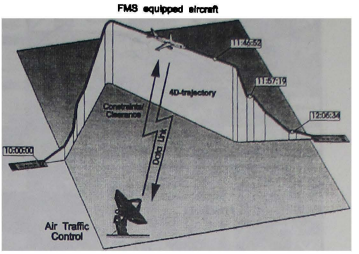


Figure 5 : Co-operative Flight Management

The main features added to the AFMS capabilities can be summarized as follows:

- Computation of 4D-trajectories on board considering
  - constraints received via data link from ATC,
  - aircraft performance parameters,
  - economical criteria, etc.
- negotiation of the flight plan with ATC/ATM by means of data link connection, and
- 4D-guidance capabilities along the negotiated trajectory.

Figure 5 shows the scenario of co-operative flight management.

The development of the AFMS was accompanied by the PHARE project "Airborne Human Machine Interface" (AHMI), where existing displays in the standard cockpit were modified in order to improve the interaction between man and machine in a first step. The Navigation Display (ND) was extended to an "Interactive ND" by the introduction of a touch pad at the arm rest of the pilot. The tactical planning was transferred to this interactive ND, which could be operated through two separate display modes:

- The PLAN mode supported flight planning and enabled the pilot to initialize and edit the constraint list representing the basis for any 4D prediction. An example for the layout is shown in Figure 6.
- The MONITOR mode supported the pilot monitoring the flight progress with respect to the active 4D-trajectory and the contract between aircraft and ATC.

The strategic planning, especially the pre-flight planning still was done with the help of a Control Display Unit (CDU), which was modified too.

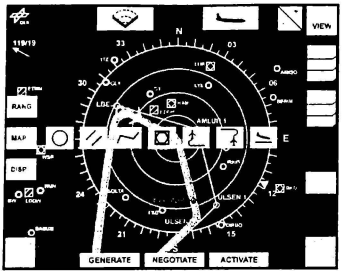


Figure 6: ND-Horizontal Display

AFMS VALIDATION

AFMS and AHMI were tested during numerous simulation runs and flight test campaigns. These tests and demonstrations were carried out within the PHARE Demonstration PD/2 [1]. The main purpose of these flight trials was to demonstrate the ability of a 4D-FMS and datalink equipped aircraft to negotiate on board predicted 4D-trajectories with an arrival planning system on the ground developed within PD/2 and then to fly the agreed trajectory within given tolerances fully automatically, i.e. without further intervention by the PD/2 ATC controllers working within the PD/2 simulation environment.

DLR's research aircraft ATTAS represented the airborne side in the PD/2 demonstrations, in which it was used as a live 4D-FMS equipped aircraft flying an approach route to Frankfurt from the western sector. The flight was integrated with other simulated traffic in the Air Traffic Management and Operations Simulator (ATMOS) hosted PD/2 arrival planning system and was handled by the controllers in the same way as the other simulated aircraft.

Aircraft control responsibility for the PD/2 flight trials was allocated to the pilot seated in the ATTAS Experimental Cockpit. The following sections briefly describe the involved experimental facilities. Later on, the results of flight testing are summarized.

ATTAS TEST AIRCRAFT

The ATTAS is one of DLR's experimental aircraft. It is a twin turbofan 44 passenger transport aircraft of type VFW 614 (see Figure 7). The aircraft is equipped with an experimental duplex Fly-by-Wire flight control system and a versatile airborne computer and sensor system comprising a duplicate set of inertial sensor systems , air data systems and navigation receivers. Two of the airborne computers are of particular interest to experimenters, since they can run software which is specific to the experiment.

During the PD/2 demonstration flights software packages were implemented which provide high precision navigation data and actual wind data. An experimental AFCS comprising autopilot and autothrottle functions was also implemented.



Figure 7: DLR's ATTAS Experimental Aircraft

ATTAS EXPERIMENTAL COCKPIT

The ATTAS can be flown in a basic mode from the right hand seat in the front cockpit by means of the conventional mechanical controls, or in a Fly-by-Wire-mode from the left hand seat in the front cockpit via side stick control. A further simulation mode allows to fly the ATTAS under instrument flight rules from the ATTAS Experimental Cockpit which is installed in the former passenger cabin directly behind the (primary) cockpit. The ATTAS Experimental Cockpit represents the right hand side of a modern transport aircraft cockpit. However, the ATTAS Experimental Cockpit may also be operated as a fixed-base cockpit in a simulation environment.

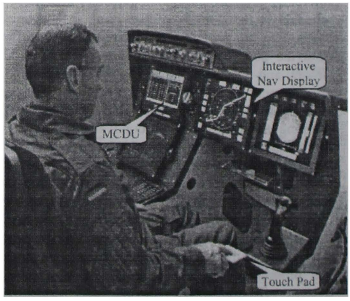


Figure 8: ATTAS Experimental Cockpit

For the PD/2 experiments the ATTAS Experimental Cockpit was equipped with pilot I/O-device hardware (refer to Figure 8 for details) such as the

- Flight Control Unit located in the glareshield panel for mode control of the experimental AFCS,
- standard 5-inch A310 EFIS displays serving as Primary Flight Display (PFD), Horizontal Situation Indicator and Engine Display, respectively, the displays being connected to programmable symbol-generators,
- the touch screen CDU of the EFMS located in front of the pilot, and the
- interactive ND realized on a 13-inch LCD display located in the central instrument panel above the center pedestal, connected to a workstation representing the display generator, and a touchpad input device driving the cursor on the display.

RESULTS OF EVALUATION

During the demonstration runs of the PD/2 demonstration week ATTAS was operated as a live 4D-FMS aircraft. The live aircraft was handled by the controllers in the same way as the other simulated aircraft, but with the trajectory negotiation process being conducted directly between the planning controller working with the PD/2 ground system and the EFMS installed in the aircraft. The communication between the PD/2 ground system and the EFMS was conducted over DLR's telemetry data link.

The flight tests confirmed the ability of a 4D-FMS equipped aircraft to agree conflict-free 4D-trajectories with ATC and to fly them, while operating within continuous 4D-tolerances. The demonstration yielded numerous results which are described in [4] more in detail. As the most important outcome of the experiments it could be shown, that co-operative flight guidance assisted by means of a 4D-FMS is feasible and a key approach for future ATM. It was demonstrated, that delivery accuracies of 10 seconds at a Metering Fix and 5 seconds at an Approach Gate are achievable. In addition test pilots appreciated the Airborne Human Machine Interface.

ENHANCED VISION MODULE

DLR's Enhanced Vision System serves as a second example for pilot assistance modules. EVS provides comprehensive situational awareness especially during flight phases close to terrain like approach, landing, taxiing and take-off. This is achieved by sophisticated methods of situation assessment based on data fusion techniques applied to data acquired by on-board multi-spectral sensors[5]. Figure 9 shows DLR's Enhanced Vision Concept.

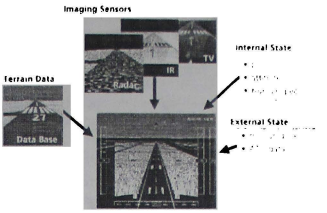


Figure 9: DLR's Enhanced Vision Concept

DLR'S ENHANCED VISION CONCEPT

Situation assessment is carried out by fusing a variety of data sources of complementary character. Most important data sources are forward looking imaging sensors mounted on-board the aircraft. Depending on the mission the sensors may cover several frequency domains from ultra-violet cameras, TV-sensors, infra-red cameras up to various radar sensors. Additional data sets are provided by the IPA's data management module and consist of

- terrain data bases,
- internal state vectors (speed, altitude, etc.),
- mission data,
- ATC data (e.g. other aircraft) and many more.

By sophisticated data fusion techniques a comprehensive representation of the aircraft's situation can be generated and displayed to the aircrew in order to achieve proper situation awareness.

The proposed Enhanced Vision concept is closely related to the well-known concepts of synthetic vision and sensor vision. These two vision concepts suffer from the restricted input data the rely on. Synthetic vision lives from terrain data bases only, which are transformed regarding the own state vector and then are displayed to the pilot possibly including additional guidance symbolism. Although the synthetic images are clearly understandable for the pilot they suffer from a lack of reliability. Data bases may not be up to date and obstacles usually are not modeled and therefore not displayed.

On the other hand sensor vision systems are referencing the real situation in front of the aircraft as they are acquiring images by means of sensors and then display those images to the pilot. Unfortunately weather independent operating sensors as for example the imaging millimeter-wave radar deliver images of a quality which is not intuitively understandable by an operator.

Enhanced Vision combines the advantages of synthetic and sensor vision avoiding their disadvantages. The combination of complementary data sources like data bases, perspective images and range images allows the computation of easily understandable images containing up-to-date information. Co-operative obstacles like other aircraft are detectable as well as non-co-operative obstacles like e.g. a lost luggage cart on the taxi way[7].

ENHANCED VISION SENSORS

It easily can be seen, that the performance of the Enhanced Vision System is strongly depending to the selection of imaging sensors. The need of obtaining images under all weather conditions may serve as an example for this effect. Therefore DLR investigated the capabilities of already available sensor systems as well as future sensor concepts carefully. A survey on EVS-capable sensors is given in [10].

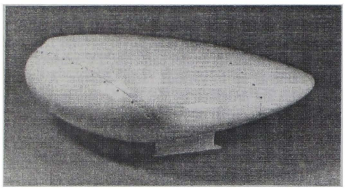


Figure 10: HiVision imaging radar (by DaimlerChrysler Aerospace)

waveform	FMCW
scanning principle	frequency scanning
center frequency	35 GHz
scanning (image) rate	16 Hz
emitted power	500 mW
resolution (angle)	0.26°
azimuth beamwidth	0.8°
azimuth coverage	41°
range resolution	6 m
detection range against a person	10 km
antenna size	86x15x30 cm^3

Table 1: HiVision Performance Characteristics

The most promising sensor technology in this field is the imaging millimeter-wave radar developed by DaimlerChrysler Aerospace. This radar is operating at 35 GHz center frequency, where the atmospheric effects are minimized. Furthermore the emitted power is continuously less than 1W, which enables the use of the radar for ground operations like taxi guidance or obstacle detection on airports without exposing humans to dangerous high frequency radiation. The radar provides 16 images per second with a range of up to 10km, which exceeds the requirements of Enhanced Vision sensors. Figure 10 and Table 1 give an impression of this promising sensor. Figure 11 shows an image acquired by the radar approaching Celle airport with additional primary flight display symbolism.

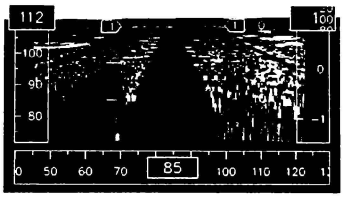


Figure 11: HiVision Image with PFD Symbolism

Besides radar technology DLR is looking into infra-red sensors for use in Enhanced Vision Systems. Although the degree of independence against adverse weather conditions compared to imaging radars is smaller, they bring a new quality of information to the data fusion process. While radars provide "range-images", infra-red cameras deliver perspective ("angle-angle-") images. Therefore infra-red and radar supplement each other in a complementary way.

EVS-SIMULATOR

DLR's Institute of Flight Guidance is performing EVS-related research covering a wide range of topics from conceptual design of Enhanced Vision Systems up to validation of EVS-elements using simulation and flight testing. In order to demonstrate the benefit of Enhanced Vision in flight simulators or test aircraft, an EVS-simulator was developed. Figure 12 shows the concept of this facility.

Core of the simulator is the data fusion processor, which merges relevant input data in order to obtain a description of the present situation. Input data are sensor images of different spectral domains as well as a terrain model data and a state vector of an aircraft. The result of the fusion process is the Enhanced Vision Display presented to the pilot. So far this simulator follows the EVS concept as described above. In order to allow the integration of this EVS-simulator in flight simulators or other experimental facilities, where real sensors cannot be used to acquire the multi-spectral images, additional sensor simulations have to be introduced.

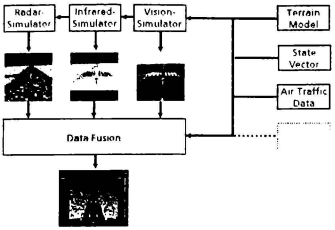


Figure 12: EVS Simulator

Within a flight simulator a pilot usually guides a simulated aircraft through a virtual landscape given by a terrain data base. In this case the state vector of the simulated aircraft and the terrain data base are used to drive sensor simulations computing images as the real sensors would provide in the real world at the corresponding situation[3]. The simulated sensor images are fed into the fusion process and the Enhanced Vision Display can be computed. The interfacing at this point is designed in a way, that the sensor simulations can be cut off and real sensor data as acquired during field or flight tests can be fed in. Therefore a very flexible tool for research and validation of EVS systems is available. As examples Figure 13 and Figure 14 show a simulated radar image and the computed corresponding Enhanced Vision Display during an approach towards runway 25 right on Frankfurt/Main airport.

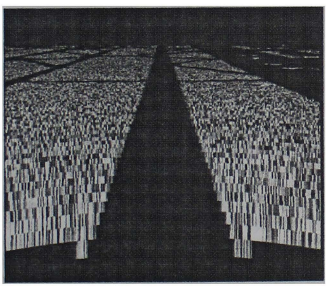


Figure 13: Simulated Radar Image

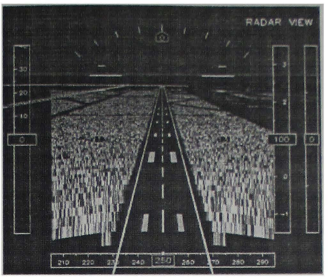


Figure 14: Simulated Enhanced Vision Display

EVS-VALIDATION UTILIZING DLR'S TEST AIRCRAFT DORNIER DO228

The presented EVS-simulator can be integrated in various types of simulation facilities and experimental equipment[4]. Within the DLR project "Advanced Visual System for Situation Awareness Enhancement" (ADVISE) DLR's test aircraft Dornier DO-228 D-CODE is being equipped with EVS sensors and the EVS simulator facility as described above. Figure 15 shows the aircraft during flight tests with the first air-worthy prototype of DASA's HiVision radar (on top of the wings).



Figure 15: DLR's DO 228 carrying EVS equipment

This flying EVS-experimental platform will be used for various validations and experiments. One powerful feature in this context is the ability of performing in-flight simulations of future sensor systems. Those novel sensors can be modeled and simulated within the already described EVS-simulator on board the aircraft and therefore be "flight tested" before real systems are being built. Furthermore different concepts of the human machine interface of Enhanced Vision can be validated in the cockpit under realistic conditions. The fully equipped aircraft will be in service mid of year 2001.

VALIDATION OF PILOT ASSISTANCE "GATE TO GATE"

Besides the validation of single assistance modules like AFMS, AHMI or EVS, first experiments with several modules integrated into an IPA gate-to-gate have been carried out in order to prove the overall concept of pilot assistance as presented above. One of these experiments was carried out end of 1999 at the Airbus A340 full flight simulator at the Berlin Flight Simulation Center ZFB.

The A340-simulator generally used for training purposes was equipped with the Advanced Flight Management System including the Airborne Human Machine Interface assisting the en-route guidance, the Enhanced Vision Display for approach and landing and a Taxi Guidance display for taxi operations. Figure 16 shows this facility from outside, Figure 17 gives an impression of the cockpit with Enhanced Vision and Taxi Guidance displays. In order to install the AFMS and AHMI modules, further displays and a touchpad as used within the ATTAS Experimental Cockpit were mounted. Figure 18 shows this installation.

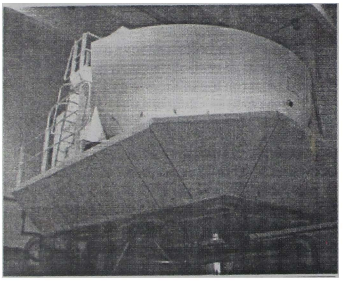


Figure 16: Airbus A-340 Full Flight Simulator

During the experiments several aircrews had to perform simulated flights from Berlin to Frankfurt with gate-to-gate assistance by the intelligent Pilot Assistant. The en-route flight guidance was carried out with the help of the Advanced Flight Management System and the Airborne Human Machine interface communicating with a simulated Air Traffic Management via data-link. Trajectories were generated on-board and negotiated with the ground-facility co-operatively, so that time-precise flights were performed. Several challenging scenarios forcing sudden modifications of the flight plan were simulated. At every time the IPA assisted the crew properly.

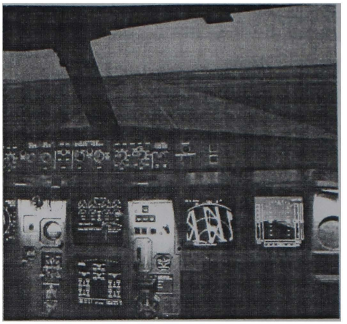


Figure 17: EVS and Taxi Guidance in A-340 Simulator

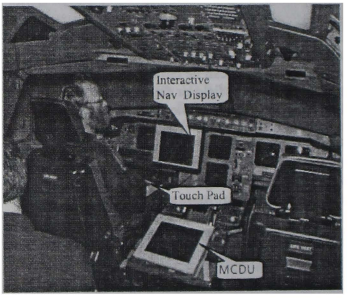


Figure 18: AFMS and AHMI equipped A-340 Simulator

During approach and landing the Enhanced Vision System provided a clear view of the upcoming scenario. Simulated obstacles on taxiways were detected independently under all weather conditions. At least, a Taxi Guidance display guided the crew safely to the destination gate.

The demonstrated concept was appreciated explicitly by the pilots. However, further research on the human machine interfaces has to be carried out in order to improve the homogenous layout of HMI.

CONCLUSION

DLR' Institute of Flight Guidance explores concepts and systems to increase safety and economy of aviation.

A promising approach in this domain is the improvement of the aircrew's situational awareness

by introducing pilot assistance systems. DLR developed an overall concept including a generic system architecture in order to provide pilot assistance during all phases of flight from "gate-to-gate". Individual assistance functions are implemented as modules grouped around a core system consisting of a central data pool and a module manager. For several tasks of flight guidance, a number of highly specialized modules was developed and tested in national and international research programs.

Beside the system architecture two assistance modules were introduced more in detail. The first module is the so-called Advanced Flight Management System (AFMS) supporting en-route flight guidance. AFMS extends the conventional flight management into a 4D-Flight Management System. Detailed trajectories are generated automatically with respect to aircraft position, altitude and time. By means of data link connection to ATC/ATM the flight plan is negotiated in a co-operative manner and an on-board module allows the 4D guidance along the negotiated trajectory. Although this process is highly automated, the pilot still is the decision making authority on board. In order to provide a sufficient interfacing between pilot and aircraft an Airborne Human Machine Interface (AHMI) was developed.

As a second assistance module, an Enhanced Vision System (EVS) was introduced, which provides comprehensive situational awareness especially during flight phases close to terrain. Enhanced vision gives the aircrew an artificial view out of the cockpit even under adverse weather conditions. Images acquired by forward looking, multi-spectral sensors are combined with terrain data bases and status data by application of data fusion techniques. The resulting description of the situation is given to the pilot by means of head up or head down displays.

The individual assistance modules were validated and tested using various modes of simulation and flight testing in order to keep an optimal balance between results and costs. Key to DLR's capability in this field is the availability of all different levels of simulation from sensor simulation as used for the development of EVS data fusion algorithms up to in-flight simulation with the Advanced Technologies Testing Aircraft System (ATTAS) for AFMS and AHMI validation.

In addition, a first evaluation of an overall pilot assistance "gate-to-gate" was carried out integrating modules for en-route navigation, approach, landing and taxiing with the generic structure and testing it on an Airbus A340 full flight simulator. Airline pilots gave a positive response to the presented concept and appreciated DLR's approach. Within further national and international projects the development of modules and the core system will be continued in order to provide a contribution to increased safety and economy of aviation.

References

1 Adam, V., Teegen, U.: PD/2 Final Report, Annex F: Airborne Aspects of PD/2, EUROCONTROL, DOC 97-10-13, 1997.

2 Adam, V., Ingle, G., Rawlings, R., Experimental Flight Management System, AGARD CP 538, 1993, Berlin

3 Doehler, H.-U.; Bollmeyer, D.: "Simulation of imaging radar for obstacle avoidance and enhanced vision" in "Enhanced and Synthetic Vision 1997", J.G. Verly (Ed.), SPIE Vol. 3088, 1997, pp. 64–73.

4 Hecker, P.; Doehler, H.-U.: "Enhanced Vision Systems: Results of Simulation–and Operational Tests" in "Enhanced and Synthetic Vision 1998", J.G. Verly (Ed.), SPIE Vol. 3364, 1998.

5 Hecker, P.; Döhler, H.-U.; Suikat, R., Enhanced Vision Meets Pilot Assistance, „Enhanced and Sythetic Vision 1999", J.G. Verly(Ed.), SPIE Vol. 3691, 1999, Orlando, USA

6 Hecker, P., Aircrew Assistance by Sophisticated Vision Systems, Concepts for Increased Flight Safety, „International Workshop On Techn. Elements for Aviation Safety", 1999, Tokyo

7 Korn, B.; Döhler, H.-U.; Hecker, P., MMW Radar Data Processing for Enhanced Vision, „Enhanced and Sythetic Vision 1999", J.G. Verly(Ed.), SPIE Vol. 3691, 1999, Orlando, USA

8 Helmke, H., Höppner, F., Suikat, R.: Generic Architecture for a Pilot Assistant System, Conference proceedings of The Human Electronic Crew: The Right Stuff?, Kreuth, Germany, 1997.

9 Onken, R.: Knowledge Based Cockpit Assistant for IFR Operations, in: Knowledge Based Systems and Applications in Guidance and Control, AGARD-CP-474, CGM Symposium, Madrid 1990, p. 14.1-14.9;

10 Rodloff, R.; Döhler, H.-U.; Hecker, P., Image Data Fusion for Enhanced Situtation Awareness, RTO-SCI-Symposium on: „The Application of Information Technologies to Mission Systems", 1998, Monterey, USA

11 Suikat, R., Helmke, H. Höppner, F., Pilotenunterstützung Gate-to-Gate als Antwort auf die zukünftigern Anforderungen im Cockpit, DGLR Jahrestagung, 1999, Berlin.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37291

AIAA-2000-4579

FLIGHT SIMULATION USING ACTUAL HELICOPTER EQUIPPED WITH FULL-AUTHORITY FBW SYSTEM

S.Ozeki, K.Kimura, T.Tomio
Advanced Technology Institute of Commuter helicopter, Ltd. (ATIC)
Kakamigahara City, Japan

ABSTRACT

Advanced Technology Institute of Commuter helicopter Ltd. (ATIC) has been conducting research and development on flight safety technologies, utilizing flight control law with full authority Fly-by-Wire (FBW) control system. ATIC makes a research helicopter and is now performing flight test to evaluate these safety technologies.

In this paper, we mainly describe the hardware-in-the-loop flight simulation test using an actual helicopter equipped with ATIC on-board system.

Through this flight simulation test, we were able to confirm the performance and reliability of our on-board system efficiently and the flight safety of our research helicopter before the flight test.

Furthermore, in this flight simulation test, pilots were able to familiarize with the new on-board system and with the recovery procedure in case of possible failures.

ABBREVIATIONS

A/D	Analog/Digital
ACAH	Attitude Command Attitude Hold
ACVH	Attitude Command Velocity Hold
ADC	Air Data Computer
AP	Auto Pilot
ASSC	Active Side Stick Controller
ATIC	Advanced Technology Institute of Commuter helicopter, Ltd.
CCDL	Cross Channel Data Link
CP	Collective Pitch
CPU	Central Processing Unit
CRT	Cathode-Ray Tube
D/A	Digital/Analog
DDV	Direct Drive Valve
DGPS	Differential GPS
DIO	Discrete Input Output
DL	Direct-Link
EMI	Electro-Magnetic Interference
FBW	Fly By Wire
FCC	Flight Control Computer
FD	Flight Director
FMC	Flight Management Computer
FMS	Flight Management System
GE	General Electric, Inc.
GPS	Global Positioning System
IFR	Instrument Flight Rules
IMC	Instrument Meteorological Conditions
IRS	Inertial Reference System
KHI	Kawasaki Heavy Industries, Ltd.
LCD	Liquid Crystal Display
MFD	Multi-Function Display
ND	Navigation Display
PFD	Primary Flight Display
RC	Rate Command
RCAH	Rate Command Attitude Hold
RPM	Revolution Per Minute
SPICE	Stick & Pedal Interface & Control Electronics
VMC	Visual Meteorological Conditions
V_{NE}	never exceed velocity

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

INTRODUCTION

Recently, accidents of helicopters have increased in Japan. One of the most important causes is an operation in bad weather condition (poor visibility). We can divide this cause into two cases, one is the loss of communications and his position during the unexpected rapid weather change, and the other is the excessive workload by the loss of visual cue.

Under the above circumstances, the drastic improvement of the helicopter safety in bad weather has been researched by ATIC since 1994.

Our goals are;

1. To improve the of instrument flight capability of helicopters.
2. To improve the IFR operation capability based on the advanced navigation systems using the Differential Global Positioning System (DGPS).

We made the research helicopter to prove these goals. At present, the flight test is being undertaken with the research helicopter.

This paper mainly describes the hardware-in-the-loop flight simulation test as a final verification step before moving to the flight test.

OUTLINE OF ATIC

ATIC was established in March 1994 with investments from the Japan Key Technology Center (organization of the Ministry of International Trade and Industry and the Ministry of Posts and Telecommunications) to address technical aspects of the helicopter flight.

There are two research themes in the ATIC - noise reduction and flight safety improvement. ATIC Research Dept. 1 is undertaking research into external noise-reducing technology, while research into helicopter flight safety technology is being conducted by ATIC Research Dept. 2.

For flight safety, many researches have been executed in the world. The helicopter IFR operation using infrastructures such as GPS system has also been considered in Japan and it will be realized in early 21st century.

Under the above circumstances, the ATIC research Dept. 2 has studied the helicopter which can ensure safety and dispatch reliability under Instrument Meteorological Conditions (IMC) by dividing the goal into two of "simplification of the navigation" and "simplification of the flight".

Schedule of out study is shown on table 1. The concept study was carried out in 1994, and the piloted flight simulation tests were performed 8 times from 1995 to 1997. The ground test was carried out 1998 using actual hardware and software for the flight test. Flight test has been conducted since 1999 to demonstrate the outcome of our studies.

We briefly introduce the ATIC system and the research helicopter, then describe the hardware-in-the-loop flight simulation test which was conducted with the actual helicopter.

Table 1 Research schedule

94	95 96 97	98	99 2000
Concept Study	Design and Development (Including Piloted Flight Simulation Test)	Ground Test	Flight Test

CONFIGURATION OF THE ATIC SYSTEM

Figure 1 shows the configuration of the on-board system. The characteristics of the on-board system are as follows

1. Full authority FBW control system
2. Active Side Stick Controllers(ASSC)
3. Cockpit display system (PFD and ND)
4. Flight Management System(FMS) for AP and FD flight

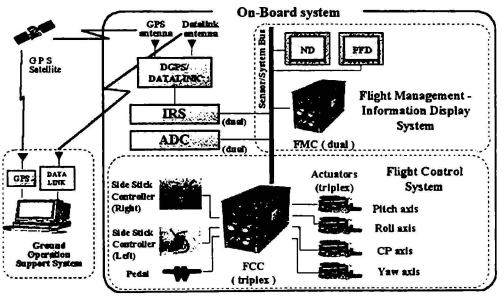


Fig.1 Configuration of the on-board system

5. DDV actuators
6. Utilization of GPS and DGPS
7. Asynchronous triplex FCCs + analog back up system
8. Dual CPUs for each FCC + CCDL between FCCs

The flight control to system is used to improve the flying qualities and simplify the control operation. This system utilizes a full-authority FBW control system to achieve both the response required in Visual Meteorological Conditions (VMC) and high stability required for instrument flights under Instrument Meteorological Conditions (IMC).

We can thus adopt sophisticated flight control laws, which enable great improvement in flying qualities. There are three types of flight control laws in FCC, RCAH(Rate Command Attitude Hold), ACAH(Attitude Command Attitude Hold), ACVH(Attitude Command Velocity Hold).

The flight management - information display system guides the helicopter and helps pilot's judgments. The flight management computer guides the helicopter using positional information obtained from the Global Positioning System (GPS) receiver. In addition, during take-off and landing, the Differential GPS (DGPS) offers more accurate positional information by using correction information obtained through a data link. With this information, a helicopter can track an accurate flight path that makes the best use of the helicopter's capabilities, such as a curved approach and landing path. The automatic flight is realized by coupling FMC with FCC.

Autopilot Couple Function realizes automatic guidance control according to autopilot commands calculated in FMC to follow flight path defined in the flight plan. Among three control modes of flight control law, ACAH is used in automatic flight and FD flight because it is most stable in these control modes. The guidance commands calculated in FMC are transmitted to FCC and are transformed to the control commands. They are also transferred to SPICE for backdrive, and the side stick moves while automatic flight in the same way as the pilot's operations. So, the pilot can intuitively comprehend the condition of the AP control. And the pilot can always take over the automatic flight by applying force to the ASSCs.

The information display function helps the pilot make decisions by displaying information required during flight in integrated and optimized images using two Multi-Function Display (MFD) units. Thus, the flight control system and the flight management - information display system together reduce the pilot workload in piloting and other non-piloting operations.

RESEARCH HELICOPTER

Figure 2 shows a picture of the research helicopter. (Components were installed on a medium sized, twin-engine helicopter, BK117.)



Fig.2 Research helicopter

The right side of the cockpit was modified to accommodate the ASSCs and two MFD's. The pilot in the right seat serves as an evaluation pilot. The left side of the cockpit is essentially unmodified. The conventional flight control system was remained for left sided pilot. The pilot in the left seat serves as a safety pilot.

Especially, as for the safety, we designed a research helicopter which had the same flight safety as BK117 because the safety pilot can disengage FBW and take over control at once when FBW system fails.

Figure 3 shows the outline of clutch and control system. When the FBW system is engaged, the conventional flight control system is connected in parallel with the FBW flight control system by using the clutch unit. The movement of the FBW actuators is transferred to the conventional flight control system via clutches. The safety pilot is able to cut off the clutches by a FBW Disengage Switch which is provided on his CP lever and is able to take over control at any time. This mechanism is designed to ensure safety during the flight test.

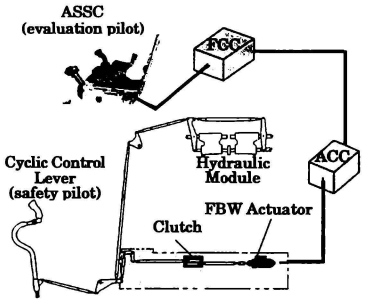


Fig.3 Clutch and control system

THE AIM OF HARDWARE-IN-THE-LOOP FLIGHT SIMULATION TEST

Ground test was conducted in the following steps.

- component qualification test
- system integration test
- total function test
  (after installation on the actual helicopter)
- Electro-Magnetic Interference(EMI) test
- hardware-in-the-loop flight simulation test

Hardware-in-the-loop flight simulation test in ground test is aimed to perform final confirmation on design before the flight test.

All the ATIC components except IRS and ADC were installed on the actual helicopter, and this research helicopter and the simulation system were connected. Therefore, we were able to carry out flight simulation test as if the pilot were actually flying by the research helicopter.

In this test environment, hardware-in-the-loop flight simulation test was conducted for the purpose of final confirmation. We confirmed that there was no problem in the function and the safety for the research helicopters system, before the flight test.

The confirmation of safety was carried out as follows.

- Safety pilot can take over the control at any time.
- Pilots can familiarize sufficiently the coordination between the evaluation pilot and the safety pilot.

This type of flight simulation test with the actual helicopter has the following advantages.

- Using the actual helicopter enables to realize the condition closer to the real flight.
- System integrity of the research helicopter can be assured.
- Pilots can be familiarized with the new system before the flight test.
- Training of coordination between the evaluation pilot and the safety pilot can be performed.
- The cost and the time schedule are saved.

CONFIGURATION OF HARDWARE-IN-THE-LOOP FLIGHT SIMULATION TEST

Figure 4 shows the set up of the hardware-in-the-loop flight simulation test.

The research helicopter was brought into the flight simulation center of Kawasaki Heavy Industries, Ltd.

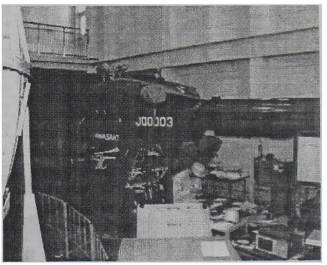


Fig.4 The set up of the hardware-in-the-loop flight simulation test

Figure 5 shows the schematic diagram of the hardware-in-the-loop flight simulation test.

The FBW control system was installed on the helicopter in the same configuration as the flight test. However, sensor signals (IRS and ADC) were substituted by simulated sensor signals calculated by the simulator computer. The communication of simulated sensor signals between FCCs and simulator computer was used the 1553B digital interface. Triplex FCCs used the pilot control input from ASSC and these simulated sensor signals. FCCs performed FBW control law calculation and

output actuator commands. The movement of the FBW actuator was transferred to the mechanical control system via clutch resulting in the movement of power actuator. The movement of power actuator was sensed and fed back to the simulator computer. In this way, the loop was closed.

In the simulator computer, the aircraft motion signals were calculated and used for generating image outside the cockpit. The generated image was displayed on three CRT displays, which were set in front of the research helicopter.

When the evaluation pilot evaluated, two units out of three units of CRT displays were installed on the evaluation pilot side, and one unit was installed on the safety pilot side. The other case, two units were installed on the safety pilot side, and one unit was installed on the evaluation pilot side.

The aircraft motion signals were also used for LCD display in cockpit, which showed the instrument symbols for the safety pilot.

Major elements of the simulator facility are as follows,

- ADI AD RTS: host computer
  - management of realtime simulation
  - calculation of flight dynamics
  - data transfer to linkage system
  - data transfer to visual system
- GE Compu-Scene IV-A: image generator computer
- CRT Visual display
- IRIS workstation: Cockpit Display computer
- Linkage
  - data transfer between host computer and cockpit
  - D/A converter, A/D converter, and DIO

Non-linear real time helicopter model was used for pilot simulation. This model features rigid blade element main rotor model , which includes flapping, lagging and RPM change degree of freedom. The accuracy of this model was verified by comparison with flight data.

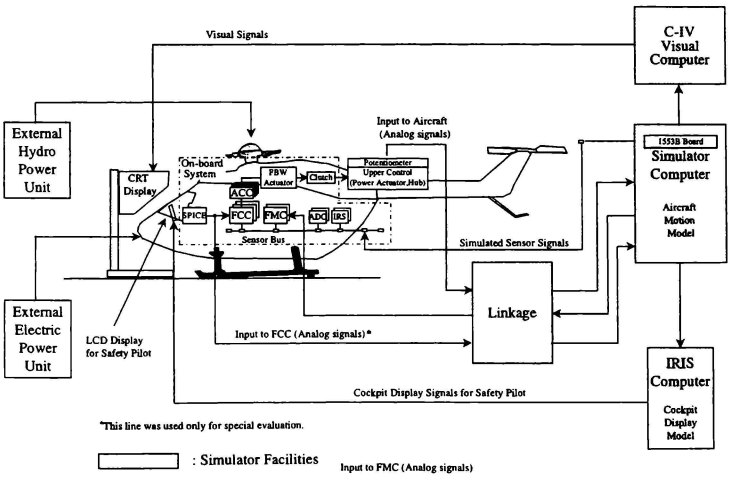


Fig.5 The schematic diagram of the hardware-in-the-loop flight simulation test

HARDWARE-IN-THE-LOOP FLIGHT SIMULATION TEST

Table 2 shows items for the hardware-in-the-loop flight simulation test. This test was carried out to confirm the function, the performance and the safety. Now, after introducing the flight control law roughly, we describe the contents and the results of the test.

Flight control law

Figure 6 shows the overall configuration of flight control law implemented in this FBW control system. As flight control law, we have designed Core Function that improves basic flying qualities and Additional Function that is added to Core Function to reduce pilot workload furthermore.

In Core Function, we aimed to realize the augmentation of aircraft stability, turn coordination and the obedient response to pilot input without harmful cross coupling between control axes. We adopted explicit model following method in which aircraft motion follows ideal motion model calculated according to pilot input. We designed three control modes with different combinations of response types. Three modes are designed as Rate Mode, Attitude Mode and Velocity Stabilization Mode. Table 3 shows the combination of response types for each axis of these control modes.

The characteristics of these control modes are

Table 2 Items of the hardware-in-the-loop flight simulation test

Evaluator	Items
Evaluation by the Engineer	Function Test of the Flight Control Law
	Function Test of the Guidance Control Law
	Function Test of the Failure Detection and Reconfiguration
Evaluation by the Evaluation Pilot	Performance Test of the FBW Control System
	Flying Qualities Test of the Flight Control Law
	Performance Test of the Guidance Conrol Law
Evaluation by the Safety Pilot	Evaluation of the Flying Qualities
	Evaluation of the Safety When Back Drive Motor Fails
	Evaluation of the Safety When Sensors Fail
	Evaluation of the Safety When FMC Fails
	Evaluation of the Safety When On-Ground Switch Fails
	Evaluation of the Safety When One Engine Goes Inoperative
	Evaluation of the Safety when Actuator Fails in Limited Authority

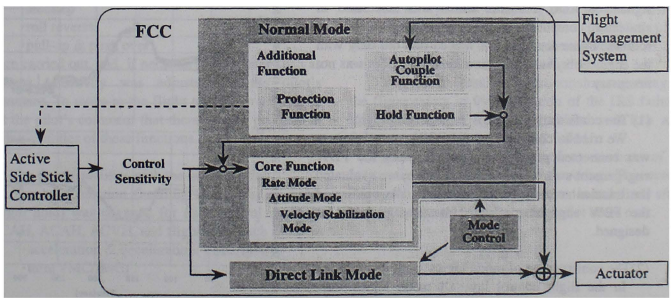


Fig.6 Configuration of flight control law

different. Rate Mode is designed to realize quick response in VMC. Attitude Mode has sufficient stability required in IMC. Velocity Stabilization Mode has especially stable characteristics for IMC to augment the velocity stabilization.

Direct-Link Mode has the same characteristics as the mechanical link. It is intended to be used as a back up when the sensor signal is unavailable.

For Additional Function, Hold Function, Autopilot Couple Function, and Protection Function are provided.

Hold function maintains flight conditions. We designed three hold modes. They are Height Hold, Heading Hold, and Position Hold.

Autopilot Couple Function realizes automatic guidance control according to the autopilot commands calculated in FMC to follow flight path defined in the flight plan.

Protection Function is aimed to notify the pilot when he is going out of flight limitations such as V_{NE}, maximum engine torque, and settling with power. Protection Function always monitors aircraft speed, engine torque, and rate of descent. And if one of these values is going out of its limitation calculated in FMC, this function notifies the pilot by shaking ASSC. V_{NE} protection provided by shaking right ASSC, Torque protection provided by shaking left ASSC, Settling with power protection provided by shaking both ASSCs.

The function test of flight control law

In the function test, It was confirmed that the movement of the flight control law inside FCC was right where it was implemented on the research helicopter. In this function test, we used simulated signals of ADC and IRS which was calculated in simulator computer. Confirmed contents were as follows. These confirmation was carried out only with the engineer because the operation by the pilot was not necessary.

(1) The confirmation of FBW engagement logic

We tried to confirm that the FBW control system was connected properly via clutch when the FBW engagement switch was operated. Also, we evaluated the transient motion on engagement. We confirmed the FBW engagement logic functions properly as designed.

(2)The confirmation of mode logics

In the flight control law, all mode logics were evaluated such as the core function mode logic, the hold function mode logic,and the mode logic for the trim function and the back drive function of ASSC. All these mode logics were confirmed right as designed.

Table 3 Combination of response types

Mode		Axis	Response Types	
			Low Speed	High Speed
Normal Mode	Rate	Pitch	RCAH	
		Roll	RCAH	
		Yaw	RC	
		CP	RC	
	Attitude	Pitch	ACAH	
		Roll	ACAH	
		Yaw	RC	
		CP	RC	
	Velocity stabilization	Pitch	ACVH	ACVH
		Roll	ACVH	ACAH
		Yaw	RC	RC
		CP	RC	RC
Backup Mode	Direct-Link	All	Replacement of the mechanical link with the electrical link.	

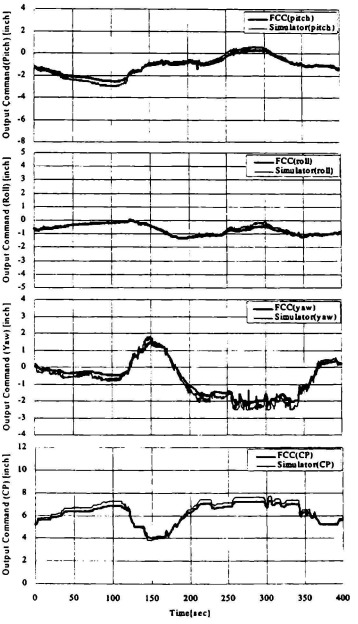


Fig.7 The comparison with FCC and simulator computer
(the acceleration-deceleration using ACAH)

(3) The confirmation of FCCs outputs

After many series of piloted flight simulation tests had been examined, the flight control law was developed. We wanted to confirm the flight control law was installed properly in FCC. To confirm this, the control input of the evaluation pilot was input to both the FCC and simulator computer with the developed flight control law at the same time. And outputs of both computers were compared. All control modes were evaluated. As a result, it was confirmed that there were few differences, which caused no problem in the calculation of the simulator computer and the calculation of FCC.

For example, Figure 7 shows the output comparison with FCC and the simulator computer when the acceleration and deceleration was done using ACAH. It can be said that two output commands are almost same. The few difference between them are owing to the time difference of input to each computer and the measuring method of these outputs.

The flying qualities test of flight control law

In the flight qualities test of the flight control law, the evaluation pilot evaluated the flight control law installed in the actual on-board system. Contents of the evaluation and the results are as follows.

(1)The flying qualities with the core function and DL

Flight tasks such as

- acceleration & deceleration(VMC/IMC)
- turn(VMC/IMC)
- climb & descent(VMC/IMC)
- descent with deceleration (IMC)
- hovering turn
- sidestep
- roll reversal
- pull-up & push over

were carried out, and, if necessary, parameter such as control sensitivity was adjusted by the pilot's comment. To move to the flight test, we were able to get the pilot's comment that there was no problem in flying qualities of these functions.

(2) The flying qualities with the hold function

The hold function (heading hold, position hold, height hold) was engaged for each control mode of RCAH, ACAH, ACVH, and flight tasks such as

- acceleration & deceleration(VMC/IMC)
- turn(VMC/IMC)
- climb & descent(VMC/IMC)
- descent with deceleration (IMC)
- hovering turn
- sidestep

were carried out, and, if necessary, parameter was adjusted by the pilot's comment. To move to the flight test, we were able to get the pilot's comment that there was no problem in flying quality about all the hold functions. We also confirmed that the hold functions were effective to reduce workload in IMC

(3) The movement of the protection function

The flights which exceeds the protection limitation of V_{NE}, maximum engine torque, and settling with power were carried out. Parameters such as timing to notify an alarm by shaking ASSC were adjusted. To move to the flight test, we were able to get the pilot's comment that there was no problem about all the protection functions.

The evaluation of safety in the sensor failure

The following procedure was carried out in the safety evaluation of the sensor failure. The evaluation pilot and the safety pilot sat on each seat in cockpit. And the simulated failure of the sensor was made to occur when the evaluation pilot flew in the same way as the actual flight with FBW engaged. When simulated sensor failure was applied, the FBW system detected the failure, then illuminated the caution light. Upon illumination of the light, the safety pilot immediately took over the control.

This evaluation was conducted on the assumption that the worst sensor failure(IRS failure or ADC failure) occurred at hovering or at high speed(near V_{NE}), and it was confirmed that there was no problem in flight safety when the safety pilot took over. After a safety pilot got fully familiarized, the safety pilot learned to identify failure by the movement of the control stick despite the caution light.

Table 4 is the result of questionnaires regarding to the flight safety in the evaluation of the IRS failure. This result shows that the workload was low and there was no problem when IRS failed.

Figure 8 shows the transient motion of a helicopter when IRS failed at hovering. This result shows that the attitude of helicopter is stabilized after about 10 seconds from the safety pilot takes over.

Table 4 Evaluation result of the safety pilot when IRS fails

	Take Over by Safety Pilot	
	IRS Failure at hovering	IRS Failure near V_{NE}
Pilot A	2	2
Pilot B	2	2
Pilot C	2	2
Pilot D	2	2
Pilot E	3	2

1:Pilot workload is very low
and there is no problem for flight safety.
2:Pilot workload is low
and the workload does not influence flight safety.
3:Pilot workload is high and the workload
probably dose not influence flight safety.
4:Pilot workload is very high
and the workload influence flight safety.

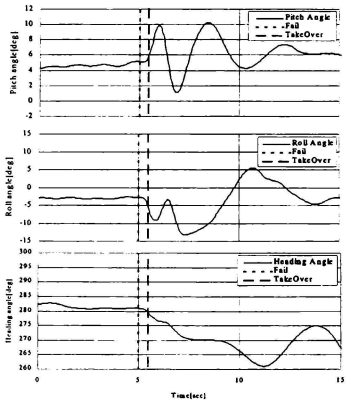


Fig.8 The transient motion
when IRS failed at hovering

CONCLUSIONS

Through the hardware-in-the-loop flight simulation test, we confirmed as follows.

1. We were able to confirm all functions with all systems integrated.
2. It was confirmed that there was no problem in the flying qualities and the guidance performance with all systems integrated. Furthermore, the evaluation pilot was able to be fully familiarized with the flight control law, the indication, the operation of MFD and CDU, and so on.
3. We confirmed that a safety pilot could take over safely.
4. The pilot learned to respond properly when the system failed. This familiarization enhances the safety in the flight test.

We are now in the halfway of flight test phase. We are going to perform flight test about 100 flights.

Through flight test, we will evaluate the improvement of the ability for instrument flight and the ability for automatic flight and FD flight using DGPS.

REFERENCES

1. I.Sudo, Y.Kubo, T.Taira, and S.Tanase, "DEVELOPMENT OF FLIGHT CONTROL LAW FOR CIVIL HELICOPTER", Heli Japan 98, T9-2, pp387-396, April 1998.

2. Koji Amano, Takaaki Taira, Shoji Tanase, Ikuo Sudo, Shuichiro Ozeki, "Study to Improve Flying Qualities in IMC for Civil Helicopter", the 36th Aircraft Symposium of JSAS, Oct. 1998

3. Y.Kumamoto, H.Fujimoto, T.Amano, K,Kobayashi, "A Study of the Flight Safety under IMC", the 25th European Rotorcraft Forum, Sep. 1999.

4. Keiji Kobayashi, Yuichi Kumamoto, Hajime Fujimoto, Takaki Amano, Kazuhiro Kimura, "Development and Evaluation of Precision Approach Profile in IMC for Civil Rotorcraft", AHS 56th Annual Forum, May 2000

5. Takeo Ito, Yoshiharu Kubo, Ikuo Sudoh, Hideki Shirasaya, Takaaki Taira, Kenzo Ikeuchi, Takeshi Tomio, "Hazard Analysis and Hardware-in-the-Loop Flight Simulation Test of FBW Research Helicopter", AHS 56th Annual Forum, May 2000

6. Koji Amano, Kazuhiro Kimura, Shoji Tanase, Ikuo Sudo, Shuichiro Ozeki, "A Study to Improve Flying Qualities in IMC for Civil Helicopters" , AHS 56th Annual Forum, May 2000

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37292

AIAA-2000-4580

RECENT EXPERIENCE AND TRENDS IN UNITED STATES AIR FORCE IN-FLIGHT SIMULATION

Norman C. Weingarten*
Veridian Engineering
Buffalo NY

Steven R. Markman
Air Force Research Laboratory
Wright-Patterson AFB OH

*Senior Member, AIAA

ABSTRACT

The United States Air Force owns two in-flight simulator aircraft, the Variable Stability In-Flight Simulator Test Aircraft (VISTA) NF-16D and the Total In-Flight Simulator (TIFS) NC-131H. Veridian Engineering of Buffalo NY operates these highly-modified aircraft. Recent programs completed on the VISTA include simulations of the X-38 Crew Return Vehicle and the Lockheed Joint Strike Fighter, and specialized training at the Air Force Test Pilot School. The TIFS has been used in recent years to support artificial visibility research for NASA's High Speed Research program. Extensive modifications were made to evaluate landings performed using high-resolution video displays of digital terrain pictures. Overviews and results of these programs will be presented. Limited R&D budgets in recent years have forced the Air Force into a new operating scheme in order to keep these unique research assets available for use. The transfer of the VISTA and TIFS to new operating schemes will be discussed.

Recent In-Flight Simulation Programs

Though flight activity has been significantly reduced in the past few years, there has been significant research and development activity on the TIFS and VISTA aircraft. These programs have included pure flying qualities research projects, systems development, test pilot training, and new aircraft development. The following is brief description of each aircraft and a discussion of flight programs which have been conducted in each of these aircraft over the past four years.

TIFS

General Description

The Air Force Research Laboratory's TIFS aircraft pictured in Figure 1 is an NC-131H (commercial Convair 580) twin turboprop transport modified as a six degree-of-freedom in-flight simulator. The TIFS aircraft is operated under contract for the USAF by Veridian Engineering, and is maintained and operated by the Veridian Flight Research Group in Buffalo, NY. The TIFS aircraft provides in-flight simulation capabilities for advanced flying qualities and display research and has been used to demonstrate advanced flight control concepts and avionics systems to test pilots and engineers. The TIFS aircraft also can function as an avionics flying test bed.

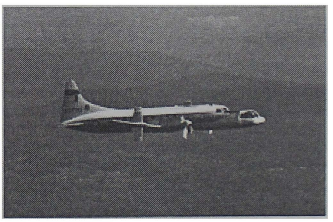


Figure 1 Total In-Flight Simulator (TIFS)

TIFS is equipped with a separate evaluation cockpit (forward and below the standard cockpit), high bandwidth electrohydraulic actuators, programmable feel systems, electro-mechanical servos for throttle control, additional control surfaces for 6 Degree-Of-Freedom (6-DOF) motion capability, programmable

This paper is declared the work of the U.S. Government and not subject to copyright pretection in the United States

displays, and the onboard computers and electronics used for the variable stability system (VSS). The additional aerodynamic controls are all-moving Side-Force Surfaces (SFS) on the mid positions of the wings, and Direct Lift Flaps which are outboard of the engine nacelles. These surfaces, the conventional C-131 flight control surfaces, the throttle servos, and the model-following system provide full 6-DOF control (rotational: pitch, roll and yaw; translational: normal, axial, and side forces) to completely duplicate the computed responses of the simulated aircraft. Figure 2 shows the layout of the TIFS aircraft for in-flight simulation.

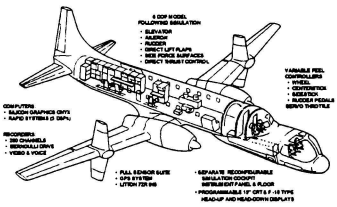


Figure 2 TIFS General Arrangement

The TIFS aircraft has been used for many research and development programs during its history. Numerous handling qualities studies have been completed on the aircraft, leading to improvements in criteria, specifications, and the understanding of airplane-pilot interaction. TIFS supported the Space Shuttle Orbiters in several development and modification programs. Military airplane development programs such as the B-1, the B-2, Tacit Blue, the X-29 and the YF-23 have used TIFS' unique capabilities for flight control system development and training prior to first flight. Advanced commercial aircraft also have been developed using the TIFS aircraft. Several supersonic transport aircraft and "million-pound" aircraft configuration programs for NASA and industry have employed TIFS for configuration and control system development, as well as for visibility and sensor investigations. TIFS has been used for human factors experiments on instrumentation, displays, control feel, motion cueing, and passenger ride sensitivity. The avionics test configuration of TIFS has been used for global positioning system (GPS), armament avionics, and remotely piloted vehicle development programs. It has also served as a training platform for test pilots and engineers. The breadth of these programs illustrates the flexibility of the TIFS.

NASA HSCT FCS, Controller, External Visibility System (XVS)

The TIFS has been involved with NASA and Industry's High Speed Research (HSR) and High Speed Civil Transport (HSCT) Program for over five years. Veridian has been an integral part of the development team which has utilized the unique capabilities of the TIFS to fulfill critical flight test requirements that could not be investigated in any other facility. This research includes development and evaluation of control laws, control techniques, control inceptors, and display elements needed by the pilot to operate the next-generation of supersonic vehicles in an environmentally-friendly, economical fashion. The primary areas of study included Guidance and Flight Control (flying qualities and flight control systems); and Flight Deck (displays, control inceptors, and visibility). Figure 3 shows the geometry of the TIFS overlaid on the HSCT.

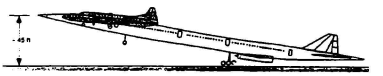


Figure 3 Sideview of TIFS/HSCT Simulation Geometry

As part of the NASA HSCT program, the TIFS simulation cockpit had to be modified extensively to accommodate the large HSCT visibility sensors and displays. The flight deck level also had to be changed to better simulate the high deck angles during approach and landing. The aft cabin area also was modified to accommodate extra sensors and computer processors. These aircraft modifications were funded under the NASA-sponsored HSCT program.

A total of 143 flights and 222.2 hours were flown during this multi-phase program. Following is a history of NASA HSCT tasks that were performed on the TIFS:

FY 96: First in-flight simulation session in TIFS. The primary area of investigation was flight control system development for approach and landing. A few flights also were flown initially to investigate the limited field-of-view associated with External Visibility Systems (XVS), a synthetic vision system for an HSCT without forward-view windows. - 23 Flights, 36.8 Hours (April - June 1996)

FY 96: An initial study was performed to determine how to modify the TIFS simulation cockpit for further HSCT and XVS flight tests.
FY 97: Second in-flight simulation session in TIFS. The primary area of investigation was pilot control inceptor (wheel/column, center-stick, side-stick) evaluations. Flight control system refinements were also made. - 28 Flights, 46.8 Hours (April - June 1997)

FY 97: The first phase of the modifications of TIFS for the XVS flight tests was performed. These modifications included a new larger-volume simulation cockpit canopy (to hold prototype XVS systems – see Figure 4), lowered flight deck, radome, and new simulation feel systems (throttle and rudder pedals). - 3 Flights, 4.8 Hours (April - May 1998)

FY 98 and FY 99: The second phase of the modifications of TIFS for the XVS flight tests was performed. These modifications included the installation of high definition television (HDTV) cameras and monitors (for forward- and in-board field-of-views), as well as a radar and other experiment-related displays and equipment in the simulation cockpit. A new SGI Onyx/2 computer and other experiment-related processors, displays, and operator interface equipment were installed in the aft cabin.



Figure 4 New TIFS Simulation Cockpit

FY 98 and FY 99: The third and fourth TIFS in-flight simulation efforts were flown in conjunction with each other. These flights were flown out of Veridian's Buffalo facility as well as NASA's Langley and Wallops Island Centers. The third TIFS effort was an investigation of further refinements to the flight control system and flexible structure (aero-servoelastic - ASE) effects on flying qualities. The fourth TIFS effort was an investigation of the initial configuration of an XVS. This XVS-1 included the integration of an HDTV forward field-of-view monitor, in-board field-of-view, radar, Global Position Satellite (GPS), Traffic Alert and Collision Avoidance System (TCAS), Automatic Dependent Surveillance-Broadcast (ADS-B), and Image-Based Object Detection (IOD) systems. Figure 5 shows the XVS-1 cockpit layout. Flight testing involved maneuvers simulating certain critical phases of flight envisioned to be experienced by HSCT aircraft during commercial operations. Maneuvers included See-to-Follow/See-to-Maneuver approaches, where TIFS flew behind the traffic aircraft at distances typical of approach to landing operations in the airport vicinity, and approach pattern entry and establishing appropriate spacing with another aircraft. Maneuvers also included See-to-Avoid/See-to-Assess Threat types of maneuvers, such as random traffic encounters where the research pilots needed to acquire and assess the potential threat the traffic aircraft was producing. In addition to work with airborne traffic, other scenarios also evaluated the ability of the pilot to see and process information from the runway environment during final approach and landing. - 49 Flights, 77.9 Hours (November 1998 - February 1999)

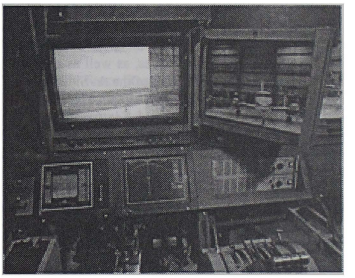


Figure 5 XVS-1 Display Layout

FY 99 and FY 00: A third phase of the modifications of the TIFS was required for the second XVS flight tests. These modifications included the installation of a three-camera HDTV camera and projector system (which replaced the single-camera HDTV monitor of the previous evaluation) in the simulation cockpit. The primary external display (PXD) consisted of three tiled HDTV video images, each produced by a rearward-facing projector, displayed on a screen directly in front of the pilot. Each of three forward-facing HDTV

cameras located in the evaluation cockpit generated one of the three tiled video images. These tiled images included a high resolution insert surrounded by two lower resolution images. Figure 6 shows this XVS-2 cockpit layout. The tiled image on the screen as well as the image on the IFOV monitor were blended with symbology overlay. The symbology overlay consisted of primary flight data, similar to typical Head-Up Display (HUD) symbology, and surveillance display symbology. The surveillance displays symbology, again the subject of experimental variation, was generated by numerous blending and detecting algorithms based on sensor inputs from the IOD, Radar, TCAS, and ADS-B subsystems. The SGI Onyx and Onyx/2 computers were used for data generation and symbology generation. Culminating this effort was a flight evaluation of this final XVS configuration. Scenarios similar to those flown in the previous XVS tests, were performed at NASA Langely and Wallops Island facilities. - 21 Flights, 36.0 Hours (August – September 1999)

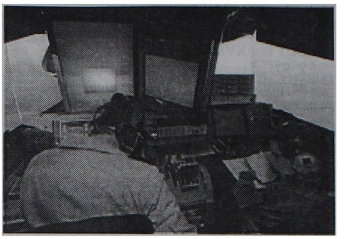


Figure 6 XVS-2 Display Layout

NASA Aviation Safety Program (AvSP) Synthetic Vision (SV)

Though the HSCT program has recently been canceled, there are areas of research and development that were associated with the HSCT that will continue. Specifically these topics relate to NASA's Aviation Safety Program (AvSP) in the areas of Synthetic Vision and other flight deck issues, as well as pilot situational awareness. The objective of the AvSP Synthetic Vision element is to provide cost-effective synthetic/enhanced vision displays using worldwide terrain data bases and Global Positioning System (GPS) navigation to eliminate visibility-induced errors for all aircraft. Synthetic Vision Systems are intended to reduce accidents by improving a pilot's situation and spatial awareness during low visibility conditions, including night and Instrument Meteorological Conditions. The types of accidents expected to be most affected by synthetic vision technologies include Controlled Flight Into Terrain, Loss of Control, and Runway Incursion accidents. Synthetic Vision Systems are characterized by their ability to represent, in an intuitive manner, the visual information and cues that a flight crew would have in daylight, Visual Meteorological Conditions. This visual information and cues are depicted based on precise positioning information within an onboard terrain database.

The TIFS, in its HSCT configuration, provided an ideal flight test vehicle to test Synthetic Vision concepts for the NASA AvSP. Towards these goals, the NASA AvSP conducted an initial investigation of Synthetic Vision (SV) systems in TIFS. These additional flights used the same XVS equipment installed for the HSCT program and were flown at terrain-impacted airports in Asheville, NC and Harrisburg, PA, where the rugged terrain was displayed on the synthetic vision system driven by cameras and computer generated imagery.



Figure 7 Synthetic Vision Display Layout

Specifically, the synthetic vision display consisted of two parts: an outside visual scene provided by a forward-looking HDTV video camera, and a computer-generated outside terrain image with flight symbology overlay. The synthetic vision cockpit layout is shown in Figure 7. One HDTV camera image was projected on the upper half of the PXD. The synthetic terrain image was projected on the lower half of the PXD. The synthetic terrain image was generated by software running in the SGI/Onyx/2 computer and used a terrain data base and GPS position. Evaluation pilots navigated in the hilly terrain around the Asheville area, as well as made approaches and actual landings using

just the synthetic image. - 19 Flights, 19.9 Hours - SV Flights (October – November 1999)

AFRL Large Aircraft Rate Limiting

Aircraft flight control system rate limiting has long been known to be a primary cause of Pilot Induced Oscillations (PIO) and possible loss of control of aircraft. This has especially been seen in fighter aircraft, such as in the loss of the Swedish JAS 39 and the crash of the YF-22. More recently, with the advent of highly augmented larger aircraft, similar rate limiting problems have started to occur, most notably, in the C-17 and Boeing 777 aircraft. The Air Force Research Laboratory (AFRL) is in the process of generating a database of configurations and test results which eventually will lead to establishing criteria dealing with such nonlinear causes of PIO. Towards these ends, AFRL funded a Test Management Project at the USAF Test Pilot School in the Spring of 1997. This flight experiment called HAVE LIMITS utilized the USAF NT-33A in-flight simulator to generate data on the effects of control system rate limiting in fighter aircraft configurations. A subsequent study utilized the TIFS to study the effects of rate limiting on PIO tendencies in large aircraft.

The objectives of this TIFS flight program were to:

- Evaluate the effects of decreasing rate limits with respect to different augmented aircraft dynamics.
- Evaluate the effects of decreasing rate limits with respect to different controller types.
- Compare the results of handling qualities evaluations using a step-and-ramp Head-Up Display (HUD) tracking task and a sum-of-sines HUD tracking task.
- Establish a database with sufficient documentation of the tasks, configurations, and test results to enable it to be used subsequently to validate criteria for nonlinear causes of PIO and to be used as a truth model for ground-based simulations to attempt to duplicate.

All configurations were flown with a conventional wheel/column and a sidestick controller. The instrument panel was set up with a head-up display (HUD) and head-down multi-function display. Each configuration was evaluated with two different tasks: a step-and-ramp HUD tracking task and a sum-of-sines HUD tracking task. These tasks were similar to the HUD tasks flown on the NT-33A HAVE LIMITS flight program, but the amplitude and bandwidth of the tracking tasks were adjusted to be appropriate for a large aircraft.

Evaluation pilots having different piloting techniques running from "low-gain" through "high-gain" were chosen. Thirty-two separate configurations were flown (2 baseline x 4 rate-limits x 2 controllers x 2 tasks) with each of the three evaluation pilots. A few of the test points were repeated resulting in 118 total test points being flown in 9 flights covering 15.6 flight-hours.

The primary result of this flight program was the generation of a well-documented, highly detailed set of flying qualities data that can be used for further research. Preliminary results also indicate the obvious fact that flying qualities decrease and PIO tendencies increase as the rate-limits were reduced. Higher gain tasks (the step-and-ramp task) expose rate-limiting-induced PIO problems more than low gain tasks. A standard wheel/column controller is more forgiving of PIO tendencies than a sidestick. And very importantly, PIO tendencies in a rate-limited configuration is highly dependent on pilot technique.

VISTA

General Description

The Air Force Research Laboratory's VISTA/NF-16D, pictured in Figures 8 and 9, is the newest in-flight simulator developed by the United States Air Force. Under a subcontract to Lockheed Martin Tactical Aircraft Systems, Veridian designed and built the programmable Variable Stability System. VISTA is maintained and operated at the Veridian Engineering Flight Research Facility in Buffalo, NY. The VISTA NF-16D aircraft provides a flight research test-bed and in-flight simulation capabilities for advanced aircraft and is also used to demonstrate advanced flight control concepts to test pilots and engineers.

The initial shakedown flights of the new VISTA aircraft began in April 1992. Shortly thereafter, the airframe was converted to a test bed for the Multi-Axis Thrust Vectoring (MATV) flight test program. This very successful test program was conducted from July 1993 through March 1994. At the completion of the MATV program, the VISTA aircraft was returned to its intended in-flight simulation configuration. VISTA became operational in January 1995 at which point it was delivered to Veridian's Flight Research Facility.

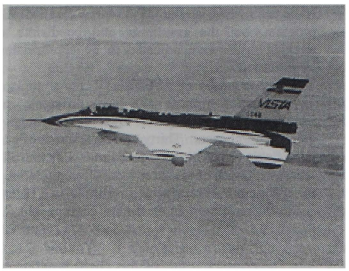


Figure 8 VISTA/NF-16D

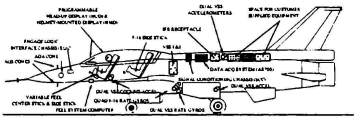


Figure 9 VISTA General Arrangement

NASA X-38 Crew Return Vehicle

The NASA Johnson Space Center is developing a prototype of the Crew Return Vehicle (CRV) to support the International Space Station requirement to return the crew to Earth in the event of a station emergency. The CRV developmental program was designated the X-38 (Figure 10). A VISTA in-flight simulation program was performed in 1998 before the second atmospheric prototype, Vehicle 132, started its drop tests of its guidance, navigation, and flight control system. The V132 was to be released from the NB-52 captive aircraft at an altitude up to 42,000 ft MSL, fly to an altitude of 23,000 ft MSL where it was to begin its transition to parafoil flight. The objectives of the VISTA testing were to:

- Validate the performance of the aerodynamics/flight control system in preparation for V132 flight,
- Generate VISTA flight data to compare against V132 ground simulations

Flight conditions to evaluate the X-38 flying qualities were selected to bound the V132 envelope. The evaluations were conducted with several simulated combinations of nominal and off-nominal X-38 V132 aerodynamics and flight controls models. Selection of these pessimistic conditions were to either establish confidence in the robustness of the control system or serve as a warning that flying qualities boundaries must be strictly observed.



Figure 10 X-38 Crew Return Vehicle

The flight control system model used for this series of tests was the X-38 GNC CSCI Version 3.6. The test conditions which simulated launch from the NB-52 were modeled by providing initial conditions to the VISTA aircraft. These initial conditions were determined from off-line simulations at NASA JSC. During an actual X-38 drop test, the V132 vehicle is released from the NB-52 mothership at approximately 0.2 g's vertical acceleration. During the first ten feet of separation, the vehicle dynamics are influenced by flow fields from the NB-52 fuselage, wing, and pylon. The V132 flight control system is not active until one second after launch. The VISTA aircraft simulated this condition starting at the point of V132 flight controls activation, i.e. one second after launch. The purpose of the VISTA simulations were to evaluate the ability of the V132 control system to damp out the motions induced by the launch.

Joint Strike Fighter (LM X-35)

The JSF (Joint Strike Fighter) CDA (Concept Demonstrator Aircraft) are multi-role fighter aircraft being developed competitively by the Lockheed Martin (LM) and the Boeing Companies for the JSF Program Office (PO). In 1998 VISTA performed an in-flight simulation of the LM JSF CDA (also referred to as the X-35, Figure 11). This simulation was used to evaluate the flying qualities of the X-35 in the Powered Approach (PA), Aerial Refueling (AR) and the Up-and-Away (UA) flight conditions in support of the control law development and evaluations. Fourteen handling quality evaluation flights (14 flight hours) were flown during June 1998. During these flights the pilots performed lateral offset approaches to touchdown and Fresnel Lens (Carrier type) approaches without

touchdowns for the PA conditions, formation and probe and drogue refueling (no fuel transfer) for the AR condition; formation and simulated air refueling tasks for the UA condition.

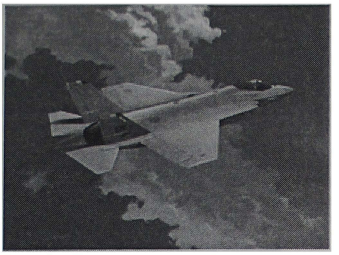


Figure 11 LM Joint Strike Fighter (JSF)

The X-35 in-flight simulation was flown from the front cockpit by the evaluation pilot (EP), utilizing the variable feel sidestick which was programmed to represent the X-35 stick characteristics. The EP stick, rudder pedal and throttle commands were sent to the Feel System Computer (FSC) which in addition to performing the stick feel computations, also contained a simulation of the X-35 flight control laws and equations of motion to simulate the X-35 airframe model. VISTA contains a GEC, 25 degree total field of view, refractive HUD with most of the navigational and weapon delivery symbology of a standard Block 40 F-16D. The HUD symbology was modified with the VISTA programmable display system to represent the characteristics of the X-35 HUD.

The offset landing task consisted of a visual approach during which the evaluation pilot aligned the aircraft approximately 300 feet to the side of the runway centerline. At 200 feet above the ground, the pilot corrected to the centerline and attempted to touch down within the desired parameters. Offsets to the left or right were used interchangeably and glide slope offsets were introduced in conjunction with a lateral offset. The offset landing task was used to introduce a planned disturbance and thus increase the workload and the direct involvement of the pilot in the manual control task near the final stages of the landing. Simulated discrete gusts and turbulence were also introduced during the offset.

Landings using the Fresnel Lens visual glideslope reference also were accomplished. A portable Fresnel lens was placed to the left of the landing aim point and was intended to be used for evaluations of the Naval variant of the X-35. Offsets of up to 50 feet left or right of the centerline also were performed. These approaches were representative of Navy Carrier approaches where no flare is completed prior to touchdown. The VISTA Safety Pilot took over at an altitude of approximately 5 to 10 feet, and performed a go around.

The probe and drogue aerial refueling task was conducted in the UA configuration using a KC-130 as the tanker aircraft. It took place at approximately 15,000 ft MSL and 210 KIAS. VISTA was flown into a loose formation position behind and to the side of the KC-130 drogue basket. This position was maintained until the variable stability system was engaged with the configuration to be evaluated. The EP varied his level of aggressiveness as required to investigate the handling qualities. The EP also attempted to maintain position (hooked up) as the tanker made a standard rate turn. In addition to these tasks, additional tasks such as wing-tip to wing-tip or wing-tip to centerline captures behind the tanker were performed.

A formation flight was also used as an evaluation task using an F-18 as flight lead. A loose wing position was maintained until the variable stability system was engaged. The EP then conducted a limited assessment of flying qualities to assure adequate control for close formation. The distance was closed to a loose fingertip formation position maintaining nose/tail separation as the lead aircraft maintained straight and level flight. This position was maintained as the lead aircraft performed moderate turns toward and away from VISTA. The evaluation was completed by allowing VISTA to deviate from the proper position vertically as well as laterally, then quickly correcting to the proper position.

Voice Recognition System (VRS)

A voice recognition system based on a Verbex commercial-off-the-shelf speech recognition system also is installed in VISTA. The system has a 500-word vocabulary that is integrated with VSS via an RS-232 serial interface. This system previously demonstrated high voice recognition rates on the order of 97% in an OV-10 test. The system has the capability for voice-prompted feedback of various types of information including fuel state, heading, and target data. The voice recognition system will allow command of digital Very High Of Boresight Seeker

(VHOBS) missile modes, radar modes, HMD/HUD, and radar and missile video.

Head-Up Display and Helmet Mounted Displays



Figure 12 VISTA Helmet Mounted Display

The VISTA Programmable Display System (PDS) is composed of fully programmable Head-Up Display (HUD) and Helmet-Mounted Display (HMD) systems. The programmable HUD uses the existing F-16 pilot display unit (25 degree field-of-view, stroke and stroke-on-raster capable). The programmable HMD uses a GEC/Marconi Viper-IV display unit installed on a modified HGU-86/P helmet shell (Figure 12). The HMD is monocular, stroke-only with a 40 degree field-of-view. A Honeywell Advanced Metal Tolerant Tracker (AMTT) is an integral part of the programmable HMD system. The outside view with symbology of both the HUD and HMD are recorded for post flight data review. Both displays are available as the 'sensor-of-interest' for all F-16 weapon systems air-to-air modes.

In Flight Refueling

The VISTA is cleared to perform dry hookups to evaluate probe/drogue type refueling with the simulation system engaged. This aerial refueling task is accomplished with a Sargent Fletcher, Inc. (SFI) 370 gallon Aerial Refueling Tank System (ART/S) mounted on either left or right wing station hard-points as shown in Figure 13. The probe/drogue refueling capability is compatible with both the KC-130 and KC-10 aircraft. In-flight simulation of the refueling task for boom/receptacle refueling research programs can be accomplished with a KC-135 tanker. Because the variable stability system is single-channel, actual hookups cannot be performed due to safety concerns, limiting the task to tracking the boom. Nose separation from the boom is required while the simulation system is engaged.



Figure 13 VISTA With Refueling Tank

Air Force Test Pilot School Instruction & Test Management Projects

The Air Force Test Pilot School (AFTPS) utilizes the VISTA to instruct students and staff and for student project test flights. These flights provide students and staff with exposure to a wide range of aircraft, flight control and display characteristics in a relatively short period of one flight. The VISTA aircraft possesses the unique capability to change its flight and display characteristics rapidly and provides the opportunity to experience the qualities of each in a back-to-back manner. The content of the flights is derived from the results of previous development programs that were flown since the aircraft was put into service in 1995. The advanced system demonstration flights and ground simulations utilize most of the features of the VISTA, including its Programmable Display System (PDS), and a Voice Recognition System (VRS). The PDS provides both a programmable Heads Up Display (HUD) and Helmet Mounted Display (HMD) that can demonstrate symbology variations as well as off bore-sight targeting. The VRS allows the evaluation pilot to control PDS functions and to request aircraft information using voice

commands without requiring looking down into the cockpit. All of these systems can be demonstrated in-flight and on the ground using VISTA's unique ground simulation capability. The tasks conducted during these demonstrations include automatic test inputs, attitude captures, all axes maneuvering, target tracking, HUD or HMD generated target tracking, formation, offset and straight in landings and air-to-ground tracking tasks. The content of these flights can be changed as school requirements change, system capabilities in the VISTA change, or as unique flight characteristics are discovered through other uses of the aircraft that would be of interest to test pilots.

In addition to the standard instruction flights, the AFTPS uses the VISTA for Test Management Projects (TMPs). These TMPs are student-conducted flight test projects that include experiment design, test conduct, analysis, and reporting. Topics for these test programs typically arise from current Air Force concerns. Over the past few years these TMPs have included the following test programs:

- HAVE CAP: This TMP validated the prediction of pilot ratings of aircraft handling qualities in the landing phase of flight. The aircraft handling qualities were predicted using the control anticipation parameter (CAP) and the bandwidth criteria. The objectives of the flight experiment were to evaluate discrepancies between the CAP and the bandwidth criteria and to evaluate the advantage of including a drop-back with the bandwidth criteria. The experiment was conducted in the Fall of 1995.

- HAVE Filter: This TMP evaluated the effects of a software rate-limiter and non-linear PIO suppression filter on the pilot command of a highly augmented fighter aircraft under rate limiting in the pitch axis with the pilot performing a Head-Up Display tracking task. The experiment was conducted in the Fall of 1998.

- HAVE Track: The objective of this TMP was to investigate the use of a flight test Head-Up Display to develop a quantitative analysis methods for predicting operational handling qualities. The experiment was conducted in the Spring of 1999.

Future Directions in USAF In-Flight Simulation

As the Air Force moved into the 1990s, new factors began to influence in-flight simulator use. Decreasing defense budgets resulted in few research programs and even fewer new aircraft to investigate. New business practices enacted throughout the Air Force required an accurate cost determination and that test sponsors pay all costs that could be attributed to their project. Expensive, unique research facilities not only had to be technically justifiable, but economically justifiable. The Air Force began to retire facilities when similar ones that met needs were available elsewhere.

While the Air Force's mission remains the defense of our country, the realities of the last decade have forced new thinking. The military shrank; lower funding ultimately translated to fewer people. The mission being the same, the remaining resources were reorganized to insure that the most is obtained from every person, every piece of equipment, and every dollar. Nothing was overlooked in this scrutiny. Nothing was sacred.

Wherever possible, civilian-oriented products are being used to satisfy military needs. Expensive, rugged items specially designed to military specifications are used only where military needs dictate such increased durability. When military-peculiar items need to be developed, opportunities are sought to share development costs with industry and simultaneously develop a civilian spin-off. This concept of "dual use" helps control costs, strengthens government and industry capabilities, and transitions military technology to the civil market where appropriate.

The Air Force can no longer afford to develop, own, and operate facilities that duplicate those already in existence, regardless whether they are government owned or civilian owned. Any duplication must be justifiable based on military need. If a service can be obtained elsewhere in a timely manner and satisfies the need, then it must be obtained from that source.

The New R&D Environment

The new environment is having a significant impact on research and development and the facilities where this research is performed. With dollars tight, the actual cost of operating facilities must be determined, then reduced wherever possible. In the past, many facilities were funded on their own to support whoever needed to use the facility. In essence, this made the facility "free" to the user. The new concept of "industrial funding" means that facilities are no longer funded. The user must hire the facility and pay all costs for performing the desired service. The impact is that only the most viable programs, those that can be defended and funded,

are performed. This is also forcing duplicate facilities to consolidate to reduce redundant layers of management. Wherever possible, the use of industry and academic facilities is encouraged if military needs can be satisfied.

The surviving facilities are becoming business-like and self-supporting. They operate as much like a private business as is allowed by public law. They seek work, reassign people with the fluctuating work load, and charge for services rendered and material and utilities consumed.

The newest concept in maintaining unique or high-value facilities is to privatize them. This can be done through leasing to industry or universities, or providing the facility to a partner through a Cooperative Research and Development Agreement. In either case, the party acquiring the facility takes on the financial risk associated with operating the facility, but is given the opportunity to make money by finding their own customers.

The Impact on USAF In-Flight Simulators

The new environment is impacting the operation of in-flight simulators. Up until the mid-1970s, the TIFS and NT-33A were completely funded by AFRL. Most projects flown up to that point were to support in-house needs, but based on priority, projects for other government agencies also were flown. This changed about 1975 when the laboratory considered terminating the program. Fortunately the program was saved, but was made to operate in a self-supporting manner. The lab provided some seed funds for a very basic level of maintenance, pilot proficiency, and system upgrades. The program management had to identify all costs for performing specific projects and the sponsor had to pay those costs in order to see their project flown. In essence, the in-flight simulation program was nearly twenty years ahead of the rest of the Air Force. By the late 1980s, this support grew to about $200,000 per year from the laboratory with an additional $500,000 per year or more from headquarters for depot-level maintenance.

By the mid-1990s, the above mentioned support funds were reduced and ultimately terminated. With support funds gone, all costs then had to be paid by sponsors as part of using the aircraft. With the TIFS and NT-33A each flown about 200 hours per year, the impact was a cost increase of over $1500 per flight hour, and a system improvement was made only when a specific project needed one and paid for it.

The problem did not stop with direct operating costs. The new environment also extended to spare parts obtained from Air Force depots. In the past, the depots were funded to provide spares and overhauled components for whoever needed them. About three quarters of the cost was absorbed by the depots, making many of the needed components "free" to the users. This changed as well; all parts now must be paid for by the user and the cost passed on to the project sponsor.

In this new environment, costs cannot be absorbed by the contractor or the AFRL. There are no hidden sources of funds to make up the differences. The full cost of performing an in-flight simulation research program must be passed to the sponsor.

Transition to Cooperative Research and Development Agreements (CRADAs)

Several times over the last ten years, Air Force leadership determined that the in-flight simulators either must be retired or transferred to a flight test center. AFRL leadership always was able to counter and keep the program going, even though annual usage on TIFS dropped steadily through the 1990s, and VISTA never flew anywhere near the annual hours as did the NT-33A. By the autumn of 1999, the decision finally was made that AFRL would divest itself of all aircraft and nearly all of its other facilities.

Initially, the aircraft were offered to Air Force test centers, NASA, and Navy test centers. However, no one wanted them nor offered to help share the cost of sustaining them for unknown but likely future uses.

Retirement of the aircraft became a distinct possibility. Veridian Engineering, who operated them on an AFRL contract, made an offer: they would take the aircraft and register and operate them as civilian experimental aircraft if the Air Force could find a way to transfer them. The AFRL program office investigated options, such as giving, selling, and leasing. None of these options appeared to be doable, either by standard disposal rules or they did not satisfy other AFRL requirements. Finally, it was determined that a CRADA could be used.

CRADAs came into existence as a result of the Stevenson-Wydler Technology Innovation Act of 1980. CRADAs mostly were used to develop some product in which both the Air Force and a private company or university (called the collaborator) each had an interest. Each partner could contribute people and equipment, but neither party paid the other for performing any work (While a CRADA is a legal contract, it is not a

procurement contract, and thus the Air Force cannot pay the collaborator to perform work). When a finished product was produced, the industry partner was free to market it or apply the technology to other products. In either case, the Air Force received a royalty that could be used by the Air Force agency that wrote the CRADA. Using a CRADA to operate an aircraft certainly was a stretch of the wording in the legislation, but was determined to be consistent with the original intent.

A year-long effort ensued to write CRADAs for the two aircraft. Within a few months, AFRL and Veridian agreed to all terms and conditions, which included duration of the agreement, annual payments and flight-hour payments, purchasing spares from Air Force depots, reporting, assumption of all financial risk, and liability. The TIFS CRADA went through extensive legal review because of the many issues involved, and was signed by both parties in November 1999. Normally, a CRADA becomes effective at this point, even if it involves the transfer of government property. However, aircraft assignments are controlled at the Air Staff level, so the CRADA had to be forwarded to the Air Force headquarters to have this action accomplished.

At this point, major policy concerns were encountered. First, the aircraft assignment office refused to assign the aircraft. Their reason was that although the CRADA instructions allow assigning property, there was no policy that covered assigning an aircraft to a CRADA. Second, the property office stated that despite the CRADA instructions allowing the assignment of property, there was no property policy that authorized this.

The aircraft assignment office eventually agreed that they would assign the aircraft when the Air Staff legal office recommended it, despite the lack of specific policy. Efforts then concentrated on drafting a supply policy that covered assigning property to a CRADA, transferring property from a contract to a CRADA, and collaborator requisitioning from depots and using Air Force bases as surrogate depots. As of May, 2000, a policy that covers these issues was drafted and is being reviewed by Air Force Materiel Command.

Plans For VISTA/NF-16D

By the Spring of 2000, the Air Force Test Pilot School became gravely concerned about the future of VISTA, particularly the chances of the CRADA operation being a commercial success. As the only regular customer, they were aware of its value as a teaching and research tool, and requested that the aircraft be transferred to the school. As the "owner", they would expand VISTA's use in the school curriculum to as much as 200 – 240 hours per year. As of May, 2000, efforts to approve a CRADA for VISTA stopped and AFRL is assisting AFTPS to insure a smooth transition.

Future In-Flight Simulation Activities

Though in-flight simulation activities have diminished in the last few years, there are still many programs that are projected to use these assets. It is anticipated that the TIFS will be involved in future flight deck and pilot situational awareness studies. These would include synthetic vision and angle-of-attack display issues. New large aircraft development programs, such as the blended-wing-body configuration, new fly-by-wire transports, and space planes are candidate programs that may use TIFS. The VISTA is scheduled to be used on a continuing basis at the AF Test Pilot School for teaching flying qualities and advanced flight control concepts, as well as demonstration of systems (HUD, HMD, VRS, etc.). VISTA is an ideal vehicle to serve as a surrogate aircraft for future risk reduction testing of Unmanned Air Vehicle (UAV) designs. The entire UAV system (simulated aircraft dynamics, guidance, navigation, flight controls, ground stations, communication links, etc.) can be simulated in a real world flight environment without risking an actual test article. Other new aircraft development efforts such as the NASA X-34 and other reusable launch vehicles and space plane configurations may utilize VISTA in developing and evaluating systems and flight dynamics. In addition, development and test of systems such as displays, sensors, and controllers are contemplated.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37293

AIAA-2000-4581

IMPROVED PROCESS FOR VALIDATION OF EFCS

Andreas Persson* and Fredrik Karlsson**

Saab AB, S-581 88 Linköping, Sweden

ABSTRACT

The Validation of an Electronic Flight Control System is an expensive and lengthy process. At Saab, the Validation mainly has been performed in simulators with hardware-in-the-loop. The process has recently been made more efficient by using model simulations in a greater extent. This paper describes how the time and cost for Validation of new Electronic Flight Control System software editions for a modern fighter aircraft can be reduced.

NOMENCLATURE

EFCS	Electronic Flight Control System
HQ	Handling Qualities
SIM	Simulation
S/W	Software

INTRODUCTION

Developing new Electronic Flight Control System software editions for modern military aircraft is a costly as well as time consuming process. A large part of both time and cost is due to the Verification and Validation of the EFCS before the first flight of the new edition. Early 1998, ideas were introduced on how to make the Validation of a new EFCS edition for the Saab-BAe Gripen aircraft more efficient. These ideas have now become a reality and have been used in the daily work for about two years.

This paper gives a brief description on how the work traditionally was performed and also the principles behind the new methodology.

At this point it is important to make clear that the meaning of the terms "Verification" and "Validation" used in this paper is some check of the software for the flight control system before the first flight with the actual software edition.

Control law process

The development of a new EFCS edition follows the so-called "Control law process" which mainly consists of three different parts: Design, Validation and Flight, test according to Figure 1.

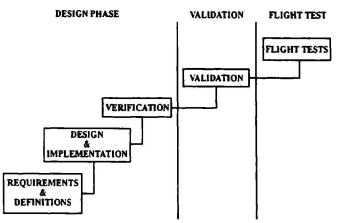


Figure 1. The Control law process

The design phase includes the definition of the edition, the actual design and implementation and finally, the Verification. The main purpose of the Verification is to check that the implemented changes fulfill the requirements and also that no undesired effects are introduced due to the changes. When the results from the Verification are such that the software edition is judged to be mature enough to be released for Validation, the final extent of the changes is established.

The Validation consists of a certain amount of different tests with the aim to give confidence for judging the airworthiness. The main purpose of the Validation

---

* MSc, Flight Control Engineer

** MSc, Flight Control Engineer

Copyright © 2000 by Saab AB, Andreas Persson, Fredrik Karlsson. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

is to give confidence on flight safety and investigate that no undesired qualities is present in the new edition, with respect to flight safety, in the actual flight envelope.

Flight tests are the final check of the new software edition in collaboration with all other systems.

Extent of the Validation

The Validation is divided into two main parts, simulations in failure free system and failure simulations.

The simulations in failure free system are performed mainly to check the handling qualities. A large number of predefined maneuvers as turn reversals, loops, landings and so forth, are performed in a number of points in the envelope with different configurations.

The purpose of the failure simulations is to check the failure accommodation in the system, and also to check the handling qualities with some failure in the system.

TRADITIONAL METHODOLOGY

The old traditional way of performing a Validation was to perform a number of simulations in a hardware-in-the-loop simulator, most often with a test pilot present, see Figure 2. This process was very time consuming and required a lot of personnel as test leaders, simulator personnel and test pilot. Most often the simulations ran over a long period of time due to fully booked simulators. Thus, a comprehensive picture of the quality of the software edition was not obtained until all of the simulations were performed.

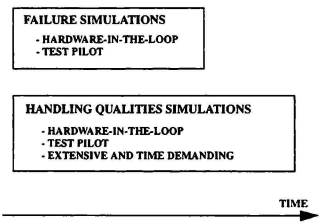


Figure 2. Traditional methodology

The evaluation of the simulations were mostly performed by manually studying the time history for a number of parameters.

Vision

The idea behind the new methodology was to decrease the time and the cost for the Validation of a new edition by increasing the use of model simulations and to introduce automatic evaluation.

NEW METHODOLOGY

By introducing the new methodology, the major part of the simulations in failure free system are performed as model simulations in an UNIX-based simulation model. Interesting parts of these simulations, e.g. simulations with a need for further investigation, may then be performed in a hardware-in-the-loop simulator with a test pilot. The failure simulations are still performed to the greater part in hardware-in-the-loop simulators, see Figure 3.

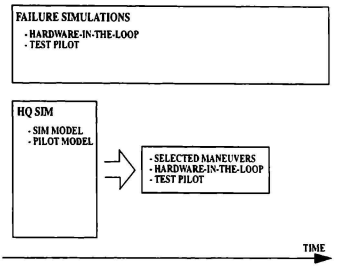


Figure 3. New methodology

Since most of the simulations now are performed in an early stage of the Validation, a more complete comprehensive picture is obtained early and problems, if any, may be found earlier compared to the traditional methodology.

The simulation model

The model used for the simulations is an UNIX-based, state space model with the possibility for the user to read and set all state parameters in all submodels. Different pilot models have been developed to be able to perform a wide range of maneuvers, as for example, loops, turn reversals and bleed off turns. This simulation model is widely used both in the design and testing of the EFCS as well as in different investigations and evaluations.

Model simulations

The model simulations are performed in non-real time automatically when the user has defined the contents of the simulations. Also some of the evaluation is performed automatically and visualized for the user.

The simulations and the evaluation is defined in control files where the user may define the actual edition, configurations, trim points, maneuvers etc. Control files for a standard Validation are defined which implies that a Validation may be run in the models whenever there is a need. The model simulation cycle is illustrated in Figure 4.

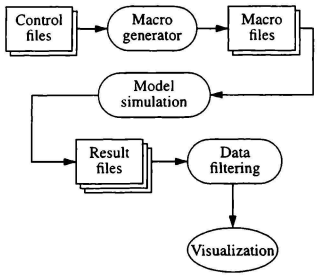


Figure 4. Model simulation cycle

The results from the model simulations are filtered and visualized for the user. E.g. maneuvers resulting in high angle of sideslip at high angle of attack may be presented for the user. Also results from other editions may be compared to the actual edition.

Visualization

Data from the model simulations may be presented in a number of different ways. During Verification and Validation maximum or minimum values of some parameters as angle of attack and load factor of primary interest. The data is mainly presented over the flight envelope. Points in the envelope where some limit is exceeded or some criteria violated is presented. See Figure 5 for an example. The user can easily define and change the limits in the visualization program. All performed simulations (maneuvers) or only selected parts, specific maneuvers or external store combinations, may be plotted.

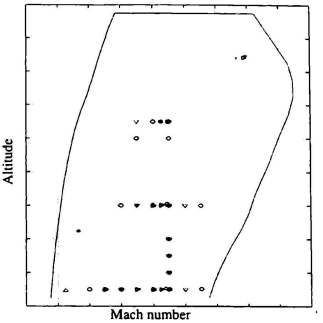


Figure 5. Points in the envelope where some limit is exceeded

The user may choose one of the points in the envelope for further investigation. It is then possible to study the time history of the selected simulation, see Figure 6. Simulation data may also be compared with earlier EFCS editions to study any desired or undesired changes of the systems characteristics. Comparison may be made both in flight envelope plot as in Figure 5, or in time history plots.

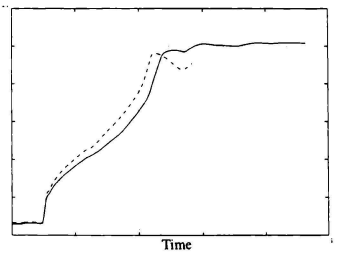


Figure 6. Time history of some parameter from one selected simulation (maneuver) with comparison between two EFCS editions.

Instead of plotting the simulation data over the envelope, it is possible to plot one parameter versus another to study any severe combinations, see Figure 7.

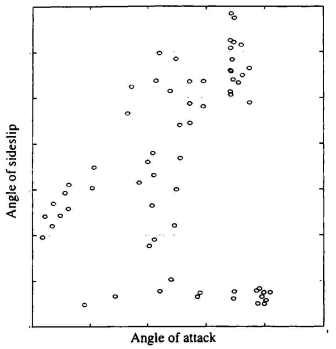


Figure 7. Angle of attack at maximum angle of sideslip

EFCS Model check out

To be able to perform simulations in a model of the EFCS, instead of as before only in the actual hardware, a special test has to be performed to verify that the model is similar to the hardware. This test is performed by "feeding" the model with the same input as for the hardware and then compare the output, see Figure 8. To be able to establish similarity, the difference between a number of parameters have to be within specified limits.

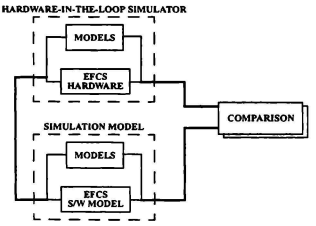


Figure 8. EFCS Model check out

CONCLUSIONS

The new process have now been used in the daily work for about two years and the change have rendered in a number of benefits:

The time required and the cost for a Validation have been reduced. The time required is judged to have been reduced with about 50 percent. The cost is somewhat difficult to judge but is in the same order as the reduced time consumption.

The test coverage is increased and thereby the quality. The number of maneuvers performed is about ten times higher.

The ability to check the similarity between two editions is much easier today due to the use of the models.

The model simulations may be performed early during the development and thereby potential problems can be discovered much earlier in the process.

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37294

AIAA-2000-4582

MODEL DEVELOPMENT OF A LEVEL D BLACK HAWK FLIGHT SIMULATOR*

Paul van Esbroeck† and Nick Giannias‡
CAE, Montreal, Canada

Abstract

While AC 120-63 has become the standard objective measure of helicopter simulator fidelity, there persists not only within the pilot community the belief that the objective tests are not a foolproof guarantor of that fidelity. This risk is particularly acute when the training regime includes aggressive flying to the limits of the flight envelope. To mitigate this risk to the Commonwealth of Australia, for the S-70A-9 Black Hawk simulator program, and to ensure fidelity throughout the entire training envelope, an extensive flight test program of over one hundred hours of flying time was proposed and executed. This paper examines CAE's involvement in the program, from testing through to model development and validation. Flight controls, engines, vibration and in particular the aerodynamic models are discussed in detail. The models, once tuned to meet the test cases required for simulator certification, elicited few negative comments in the flight regimes defined by AC120-63 objective tests. Turning flight is one area that is perhaps inappropriately and inadequately validated by AC 120-63. The existence a large objective flight test data set, beyond the limits of AC120-63, has successfully enabled the development of a high fidelity model throughout the full training envelope, reducing the need for subjective tuning, and making the tuning, where required, more efficient.

Introduction

In February of 1996, CAE was awarded a contract by the Commonwealth of Australia (CoA) to design and build a Full Flight and Mission Simulator for the S-70A-9 variant of the Sikorsky Black Hawk helicopter. The simulator was required to meet Australian CASA Level 5 and FAA Level D requirements under AC 120-63.[1] Some of the highlights of the simulator include:

(i) 200°×60° 5-channel collimated visual system with glass mirror

(ii) 2 chin window displays

(iii) 3-axis vibration platform

(iv) 6-axis motion system

(v) blade element rotor models for the main and tail rotor

(vi) NVG (simulated and stimulated) capability

The main purpose of the simulator is to provide all type conversion and type transition training for Black Hawk pilots and maintenance test pilots, as well as providing basic mission training for graduate and postgraduate candidates.

Although it has been nearly five years since the release of AC 120-63, there are still only a handful of simulators that can lay claim to fully meeting its objective test requirements. Given that helicopter simulator accreditation is still in its infancy, there is much to be learned from following the development of some of the "pilot-in-the-loop" software models.

Flight Test Program

The simulator data acquisition plan identified several key areas where flight testing was necessary to meet the technical requirements for the simulator. In particular, aerodynamics, engines, flight controls, buffets (vibrations) and sound required flight test data from a Black Hawk aircraft which was deemed to be representative of the fleet. The CoA had clearly foreseen this requirement in advance of the contract award and had set up a tasking order specifically to support flight testing for simulator data gathering purposes. This process effectively removed one of the major barriers to flight test data gathering, the provision of a suitable test aircraft.

* Copyright © 2000 by CAE Electronics Ltd., Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

† Technical Lead, Military Flight Systems

‡ Manager, Military Flight Systems, AIAA Member

Test Plan

Based on previous experience with helicopter simulator acceptance and accreditation[2], a comprehensive test plan was drawn up to ensure that (i) the requirements for accreditation data were met and (ii) the resulting simulator models were of sufficient quality to meet the training needs of the Australian Army. The latter requirement significantly broadened the scope of flight testing beyond the tests in AC 120-63.

In order to keep track of the large number of tests, a database was developed along with a graphical user interface (GUI). The system, known as *TestPlan*, was designed for use by simulator systems engineers, flight test engineers and test pilots. It provided for the creation of *test cards*, each comprising one or more *test points* and their associated test procedures, test initial conditions and aircraft configuration. Test cards could then be grouped into *flight profiles*, which represented the list of tests for a particular sortie. Any number of flight profiles could be defined and later changed, as necessary. Post-flight, the flight test engineer could then use *TestPlan* to mark off which tests had been completed as a cue to the systems engineer to review the flight test data and either accept or reject a particular test point. All tests were color-coded in the database to indicate whether a particular test card was completed and accepted, completed and rejected or not yet completed. In addition, various combinations of filters and sorting could be selected to obtain a better view of the overall test plan.

TestPlan was also used to generate briefing sheets, flight test cards and test summary sheets for use during the sortie. The briefing sheets contained detailed test information for use during the pre-flight briefing. The flight test cards provided summary information such as initial conditions, test procedures and individual test point numbers for kneepad use. The test summary sheets were used by the instrumentation engineer to follow the profile sequence and write the flight log.

Overall, the Black Hawk program was comprised of approximately 500 test cards containing over 5000 individual test points.

Aircraft Configuration

Two aircraft configurations were defined in the test plan: a clean configuration and an ESSS configuration. The ESSS configuration consisted of the aircraft fitted with the External Stores Support System and two (outboard) fuel tanks. Testing at each configuration was performed at multiple weight/cg combinations: five for the clean configuration and two in the ESSS configuration. The aircraft weight and balance envelope and ballasted limits for the clean aircraft are shown in figure 1. To ensure the weight/cg remained within the predefined limits for a particular sortie, the aircraft was refueled once the gross weight reached the minimum allowable weight for that configuration. "Hot refueling" was used to maximize flight time.

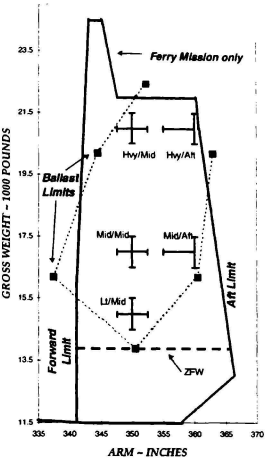


Figure 1 Aircraft weight and balance configurations used during flight test

Instrumentation

Flight test instrumentation was installed and calibrated by Australian Flight Test Services (AFTS) and personnel from the Aircraft Research and Development Unit (ARDU) of the Royal Australian Air Force.

The instrumentation system consisted of a central Data Acquisition System (DAS), DGPS receiver, video system (camera and recorder) and sound measurement system. All parameters were recorded on the DAS except for the cockpit audio/video which was recorded on a VCR and the cockpit sound which was recorded on digital audio tape. All three systems used a common IRIG time code reference.

Low speed testing was performed with the aid of a pace vehicle using a mast-mounted anemometer as a wind speed/direction reference and a fifth wheel as a ground speed reference.

A complete parameter list can be found in Table 1. In some cases, purpose mounted sensors were used in addition to recording data from the aircraft sensors, thus resulting in redundant measurements. An example of this is the aircraft air data system; while the boom data was used to compute "true" values for development of the aerodynamic model, the aircraft air data system outputs were recorded to enable more accurate simulation of the pitot static system.



Figure 2 Test aircraft

Flight Test Activities

Flight testing was conducted at ARDU's facilities in Adelaide, South Australia over the period February to September 1997. A total of 128 hours of flight testing and approximately 2 weeks of ground testing were used. The flying rate averaged 0.9 hours/day, not including the downtime experienced during a transmission change that was performed approximately 2 months into the flight test program.

Data Reduction

An extended Kalman filter (EKF) was used to reconstruct air data at low speeds using differential GPS (DGPS) and inertial data along with meteorological observations for wind data. Pressure error corrections (PEC) were developed using speed course techniques with DGPS truth data. A 6 Hz low pass FIR filter was used for post-processing the majority of analog parameters.

Model Development

Flight Controls Model

The flight control model was developed using data detailing the mechanical system as well as the flight test data collected on the aircraft. As a separate exercise to that listed in table 2., data was collected on ground using externally mounted sensors with an external hydraulic supply. In addition as part of the main data gathering exercise, the controls were instrumented using strain gauges to record forces and transducers for recording control positions, actuator (swashplate) positions and the AFCS actuator positions. Those sensors were connected to the real-time data acquisition system for in-flight data gathering.

The on-ground data was used to characterize the travel, the friction level, the force gradients and the gearing. The mechanical cross coupling functions from the mixer were also measured. Analysis of this data produced the gains, limits, and linear and non-linear functions of the control system that were incorporated into the mechanical model. The mass-damping parameters characterizing the dynamic behavior of the controls were derived using in-flight data because the hydraulic pressure supplied by the ground unit was insufficient for this task.

The major components of the flight control system such as the aft quadrant, linkages, mixer, servo actuators and boosters were modeled and integrated using these various data sources.

The resulting model was tested and the simulation results were compared against the flight test data by running integrated with the rest of the simulation models. The model parameters were then fine-tuned to achieve a match within the tolerances of the AC120-63.

Engines Model

The GE T700-701A engines installed on the Black Hawk were simulated starting from a complete fuel controller and engine partial derivative performance model from the manufacturer. The GE model covers the complete flight envelope from idle to maximum power. A flame off model was then added and the engine performance tables extended below idle to cover the startup and shutdown. The transmission and rotor brake model was then added with a generic gear mesh loss and friction.

The electric logic and indications were coded using wiring diagrams, while simulated malfunctions were added in the fuel controller, engine performance, transmission and indications with realistic effects. This engine model was then tuned to match the installed engine performance as recorded in the flight test data. These tests included, among others, hot engine shutdowns, starts with and without rotor brake, rotor speed governing in several flight regimes, and compressor wash tests.

Vibration Model

The general method of vibration model development has previously been described in detail.[3] Pilot seat longitudinal, lateral and vertical accelerations were collected on all maneuvers of the flight test program at 1000 Hz. These were then reduced into harmonic time histories (HTH). These represent the time history of each of modeled harmonics' amplitudes in each axis (that is the 1/rev, 2/rev, 1n/rev 2n/rev etc.). The harmonic time histories are then down sampled at a lower rate. The complete flight test data is then analyzed for static trimmed flight regimes. The averaged harmonics' amplitudes for the trim period, along with the average of all recorded signals, becomes

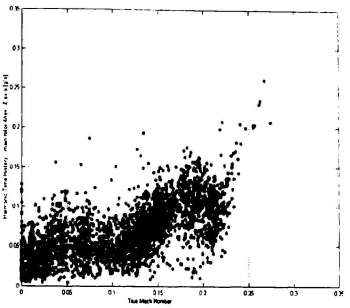


Figure 3 Vibration database 4 per rev vertical vibration levels

one element in a static trim database. The resulting static database consist of about 2500 trim points which represent steady turns, level flights, side slip, descents, climbs... covering the complete flight envelope of the helicopter. Figure 3 shows a display of the static database for the S-70A-9 helicopter 1n/rev as a function of the Mach number.

A visualization utility is used to process the database and build the driving equations for a static model based on the stepwise multiple forward regression statistical method. The error between the model and the actual vibrations is reduced to minimum by adding several multi-dimensional models until levels, trends, frequencies and axis vibration characteristics of the helicopter are covered for all the static flight regimes within the flight envelopes. For the dynamic part of the model, HTH from the aircraft data is compared with HTH outputs generated from the derived static model during the dynamic maneuver. Differences are then modeled using time dependent functions. Typically, most transitions are naturally covered by the static model. Only heavy transitions and time dependent phenomenon are not covered and need specific models. Malfunctions are modeled and tuned subjectively with the help of experienced pilot when no data is available.

Aerodynamic Model

Blade Element Rotor Model

The aerodynamic model is based on the CAE blade element rotor model.[2] It has been used on a wide variety of helicopter types including Bell 412 and 212 simulators which have been certified to AC 120-63 level C standards.

The main and tail rotors used blade element methods. The equations of motions for the six independently simulated bodies (4 main rotor blades each with flap and lag degrees of freedom, the rotor shaft with one degree of freedom, and the helicopter fuselage with 6 degrees of freedom and are solved simultaneously. Over the course of model validation additional options were added to the basic model including a dynamic wake model[4], aerodynamic phase lag[5], and virtual inertia.[6]

Aerodynamic Model Development

The shear size of the data set (a summary is included in Table 2) enabled a slightly different approach in model development than that described in reference 3. An automated method was developed to pick the best representative trims from time history data. This enabled us to convert hours of flight test data into a static trim database. The criteria for acceptable trims could be adjusted for different test types. For instance the importance of low body axis rates as a criteria for an acceptable level flight trim was relaxed to require only steady rates for banked turn trims.

Thus a database of a few thousand data points was created representing the trimmed conditions for the Black Hawk helicopter in all tested flight regimes and configurations. This database was used to check for the consistency of trends across various helicopter configurations and thus point out any idiosyncrasies in helicopter characteristics that might require extra modeling attention. Sometimes, possibly due to such factors as controls hysteresis, stabilator hysteresis, various unmeasured atmospheric effects such as updrafts or wind gradients trends will appear in certain sets of data that do not represent typical helicopter behavior. Figure 4 shows maneuvering stability trims for several airspeed and configuration combinations as a function of bank angle. The fluctuation in the data from all sources is of the order of the AC120-63 tolerances on the control position change from level trim. Unless the data set is large enough to ascertain that a particular apparent trend is an anomaly, significant effort can be put into producing this false trend in the model. At best this effort is wasted, as it adds no value to the simulation, at worst it introduces false trends into the model.

For dynamic cases no convenient method was found to convert the performance characteristics to a database. It was not practical, because of the amount of data, to validate the model against each and every case. The data set was divided into those that were required for certification, those we required for additional model development throughout the full flight envelope, and lastly maneuvers that could be used to spot check the performance of the model throughout the flight envelope. This last group included regimes and

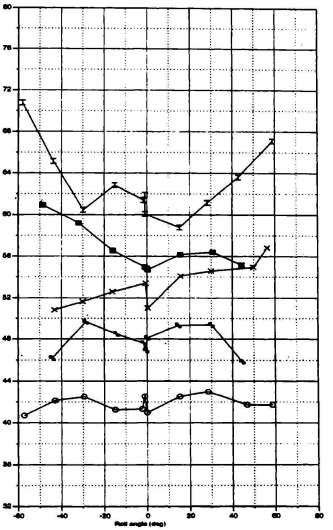


Figure 4 Maneuvering stability longitudinal cyclic data for several airspeed/configuration combinations

maneuvers that were important to the proposed training curriculum, or in areas where our helicopter models had previously elicited fidelity concerns, or in areas thought to exhibit particular traits of the Black Hawk.

The static trims database, in conjunction with some accelerations/decelerations tests was used to accurately model downwash and position error effects on the pitot tubes throughout the flight envelope. The database also proved valuable in finding some instrumentation, calibration, or data processing errors that affected some of the data.

The flight test program contained many tests not in AC120-63. We generally proceeded first in matching the requirements of the circular, looking beyond it at the early stage usually only in areas where it was essential for the model design, or where the data analysis indicated an unusual trend. While periodic subjective evaluation by pilots was still used in the development, the large array of available data has greatly reducing the amount of subjective tuning required to address performance and handling fidelity issues. When problems were found, objective data was usually available in a similar flight condition for validation and analysis. Depending on the results of the analysis, the model could be first tuned to match the objective data better before another subjective evaluation and then if necessary some subjective tuning. Often the analysis indicated no flight tuning was warranted, and thus the availability of this data bank likely prevented much unnecessary tuning of the flight model and helped to isolate the origin of a subjective handling concern as a visual, motion cueing, flight controls or auto-pilot problem. As a result only areas such as malfunction effects, where no objective data was available ended up taking a significant amount of subjective flight model tuning.

The flight test program had been designed to ensure that where possible data was gathered to ensure simulator fidelity even under abnormal operation, or with simulated malfunctions active. An example of this is flight with an incorrectly trimmed stabilator, or stabilator malfunctions. To validate these flight regimes several stabilator sweeps were performed at various speeds and aircraft configurations. This had the added benefit of allowing us to examine and then better model the effects of rotor and fuselage downwash on, and dynamic pressure effects at, the horizontal stabilator. In the past this has been an area where design data has been weak. With a small flight test data set it is often difficult to accurately determine these effects especially as within the data set stabilator incidence angles are often closely correlated with fuselage incidence angles and flight regime.

Past experience also indicated that additional data on ground under low thrust conditions would be useful. This would provide an objective means of validating the landing gear design data. This in turn would reduce subjective tuning with pilots as the gear model could more easily be implicated or eliminated as the possible source of a ground handling problem. These tests included control steps with flat pitch, as well as with low collective settings. A major revelation of the initial validation was that the simulation oleo struts, while giving the correct compressions for no thrust, were uncompressing more due to small changes in collective. In this low rotor thrust regime part of the blade is generating positive thrust, while the tip region due to blade twist is generating negative thrust. We believe that the inflow model, which depends on total rotor thrust, underestimates the local inflow at each blade element in this regime. The inflow model was altered to improve the match to flight test data. In addition ground effect modeling, which was a function of the ratio of rotor height to rotor diameter, was altered to account for an effective rotor diameter. In this regime only a portion of the rotor blade is engaged in building up the ground cushion while the outer portion of the blade is actively depleting it. The net result is that for low speed taxi and other low thrust regimes the oleos now match flight test data, are more compressed and hence stiffer than they would be without these adjustments.

An area that has generated much difficulty in the past for certain rotors is the pitch to roll coupling. As shown by others[7] and shown in figure 5, the off axis response is sometimes exaggerated by more than 1 order of magnitude, and in some flight regimes out of phase with aircraft response. While the basic model predicted static control positions quite well, on dynamic pitch inputs the initial trend was sometimes wrong and often exaggerated. By examining the roll acceleration of the flight test aircraft and comparing these with what was generated by the model, it was only during the tip path plane transition following the input that the model generated appreciable error. This was typically from control input to about one quarter of a second after control step. All inflow models tested gave a similar initial response. The onset was deemed to be too quick

following the input, to be caused by dynamic inflow effects.

An altered control phasing or an aerodynamic phasing can weaken and in some cases suppress the incorrect trend. To eliminate it entirely from the roll rate trace required exaggerated phasing and it was at the expense of a weakened on axis response and an exaggerated trend when well beyond the initial input. These secondary effects could in turn be mitigated by scaling the on axis inputs (similar to what is described in reference 7) and by introducing separate longitudinal and lateral dynamic inflow effects (analogous to those described in reference 8).

Various methods, some of which have been described above, were attempted to improve the off axis match within the constraints of real time simulation. The best

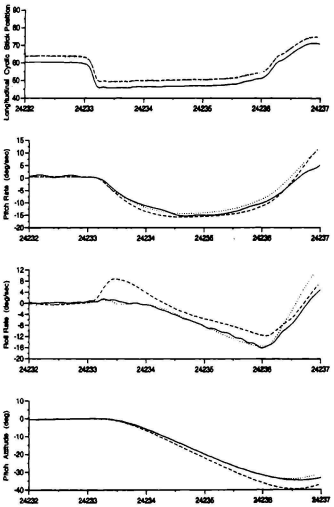


Figure 5 Pitch respose at 80 knots of basic (dashed line) and tuned (dotted line) model versus flight test data (solid line)

results were obtained by a combination of aerodynamic phase lag and the separation the swashplate longitudinal input into a high and low frequency regime and giving a different control phasing to the high pass input. The low pass was generated using a first order low pass filter with a cutoff frequency of 16 radians per second. This method (by design) did not effect the static and low frequency characteristics of the model. Longitudinal frequency sweep response tests show significant improvement in the amplitude of the off axis response. At higher frequencies the phase of the off axis response is still not correct. It is suspected that a flexible blade model would be required to improve this region. Subjective comments have been good, probably because it is initial trends that pilots tend to note, and at higher frequency the amplitudes of the off axis response are already reduced.

Spiral stability matching with Automatic Flight Control System (AFCS) disengaged proved difficult primarily because of repeatability problems. The model exhibited the correct aircraft trend. It was not divergent but was not consistently within AC120-63 tolerances. Figure 6 shows several simulation responses to a hands off release from mid speed straight and level trim. Attitude stabilization is inactive, but rate damping is still active. Unlike most fixed wing aircraft longitudinal and lateral directional responses are strongly coupled and speed control is essential for repeatable lateral behavior to within the tolerances. If the response from a level trim is not repeatable, then neither will the response when the controls are returned to the trim position after first rolling to 30 degrees bank. In speaking to both customer and company pilots, they could cite no helicopter where they would expect bank angle to remain within two degrees of wings level after 20 seconds of hands off flight. While models can be made artificially more stable so as to meet this certification requirement, I feel it adds no value to the simulation. Placing the tolerance on the lateral control change from trim required for a 30 degree turn in both directions would perhaps be a better measure of spiral stability. For the augmented case, on the black hawk, the existing tolerance is trivial as it requires only that the simulated aircraft recovers level attitude by 20 seconds. The aircraft normally recovers within 5 seconds of the release and then maintains level attitude.

In general turning flight does not have many validation tolerances and often, as in the case of the black hawk elicits a significant portion of all handling related comments, even after certification tolerances are met. These have involved the attitude required to maintain

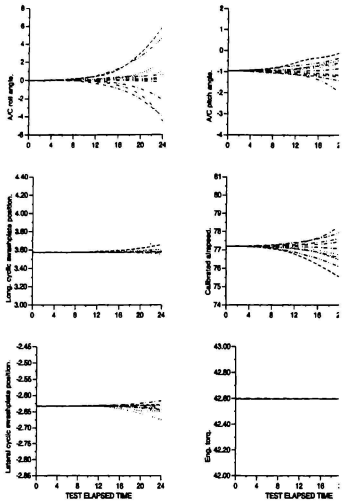


Figure 6 Hands off simulation releases from a straight and level trim

airspeed, the torque required to remain altitude, and the effect of g-loading on roll authority.

Turns, especially steep turns are an important part of many training scenarios, and aircraft behavior (pitch attitude, torque, and control deflections) are often very dependant on the direction of the turn, the configuration of the aircraft and the airspeed. I am hesitant to ask for additional tests based on our experience with military aircraft where the fidelity requirement is not limited by AC120-63 anyway but by the training requirements. The number of test points required is perhaps already quite large for commercial purposes. How could AC120-63 be improved to ensure to be a better indicator of simulation fidelity, without significantly altering the number of tests? I would recommend replacing the existing maneuvering stability, and spiral stability tests with slip ball centered constant collective and constant altitude turns in both directions, with tolerances on control position changes, pitch attitude and sideslip, torque and rate of climb changes from the initial straight and level trim.

Summary & Conclusions

The existence of a large data set, with engines, controls, swashplate and AFCS parameters, has enabled us to examine a variety of pilot comments and often quickly determine which aircraft system if any is a source of the problem. If the aircraft simulated systems show good matches to flight test performance more attention is then given to visual and motion cueing to determine if these are giving the pilot insufficient or false cues. This has greatly reduced the subjective tuning required especially for the aerodynamic and vibration models. This is only in part because the large data set has enabled the models to be checked throughout a wider envelope before the start of subjective evaluation. Once subjective evaluation has started and it has been determined that an effect is missing, (and invariably the engineers have not found all the idiosyncrasies) the data has provided objective guidelines for the subjective tuning exercise that has reduced the number of iterations required to complete the tuning process. It has also increased the confidence of both engineers and pilots in the solutions offered. The failure in the past of subjective tuning exercises in regions of the flight envelope where little data is available is often due to the inability to isolate the source of the problem with confidence. The more systems that can be shown to be correct and hence removed from the problem equation, the greater the attention that can be focused on the remaining suspects, and the less probable that the ultimate fix is in the wrong place with deleterious consequences.

The black hawk data set has revealed some weaknesses in our pre-existing blade element model. Corrections and/or mitigating factors are proposed for certain weaknesses, which may be applicable to other helicopter models.

Spiral stability requirements of AC120-63 with stability augmentation off appear to border on the limit of helicopter repeatability. Collective fixed, as well as constant altitude turns in both directions are proposed as an alternative to the existing maneuvering and spiral stability tests, as a more appropriate means of validating the spiral stability of the helicopter model, and the steady turning flight regime in general.

Objective validation cannot eliminate the need for subjective assessment. Not only to ensure the apparent seamlessness of the simulation throughout the entire flight envelope, but also to ensure that perceptual cueing is appropriate.

While subjective validation is an essential check it cannot usually point efficiently to the source of the problems it reveals. An abundance of good quality objective data from a single source is an important component in ensuring effective resolution of problems and ultimately high fidelity simulation.

Acknowledgements

The authors would like to thank the following individuals for their contribution to this program: Vito Longo, Luminita Ilinca-Ignat, Sebastien Morisset, Eric Gagnon, Marc Kyriaco, Daniel Spira, CAE; Dave Fawcett, Pete Harris, ARDU; Col White, AFTS.

References

[1] U.S. Department of Transportation, Federal Aviation Administration Advisory Circular, "Helicopter Simulator Qualification" AC120-62, October 1994.

[2] Longo, V.: "Helicopter Simulator Qualification to AC 120-63 Standards: Lessons Learned", Royal Aeronautical Society Conference Proceedings, 19 May 1999.

[3] Sabourin, M.O.: "Development and Harmonization of Motion System, Vibration Platform and Dynamic Seat Cues for Helicopter Training Simulators", Royal Aeronautical Society Conference Proceedings, 19 May 1999.

[4] Krothapalli, K. R., Prasad, J. V. R., Peters, D. A., "A Generalized Dynamic Wake Model with Wake Distortion Effects" American Helicopter Society 54th Annual Forum Proceedings, May 1998.

[5] Mansur M. H., Tischler, M. B., "An Emperical Correction Method for Improving Off-Axes Response Prediction in Component Type Flight Mechanics Helicopter Models", AGARD Conference Proceedings 592, Advances in Rotorcraft Technology, May 1996.

[6] von Grunhagen, W., "Dynamic Inflow Modeling for Helicopter Rotors and Its Influence on the Prediction of Crosscoupling". Presented at the American Helicopter Society Aeromechanics Specialists Conference, Fairfield County CT, October 11-13, 1995.

[7] Fletcher, J. W., "A Model Structure for the Identification of Linear Models of the UH-60 Helicopter in Hover and Forward Flight", NASA Technical Memorandum 110362, August 1995.

[8] Smith S., J., "Helicopter Simulation Modeling Techniques for Meeting FAA AC120-63 Level D qualification Requirements. Presented at the American Helicopter Society 56th Annual Forum, Virginia Beach, Virginia, May 2-4 2000.

Table 1 **Flight test parameter list**

Airspeed No 1 & 2
Barometric Altitude, Altitude Rate
Doppler Forward, Lateral Velocity
Pitch Attitude No 1 & 2
Radar Altitude
Roll Attitude No 1 & 2
Heading
AFCS Release Switch – Cyclic
Beep AFT, FWD, Left, Right Switch – Cyclic
Collective Command (SAS 2) HI, LO
Collective Stick Position No 1, 2 (POT)
Collective Trim Control
Collective Trim Drive Command
Collective Trim Position (Feedback)
Collective Trim Power
Collective Trim Release Switch Pilot, Co-Pilot
Cyclic Trim Release Pilot, Co-Pilot
Depart Select Switch
Engine Torque Input No 1,2 HI, LO (FM ECU)
Engine Torque Output No 1,2 Out LO, HI
HDG Slew Left, Right
Hover Trim Lateral, Longitudinal (CREW)
Lateral ACCEL No 1 (FWD RH CABIN)
Lateral ACCEL No 2 (FWD RH CABIN)
Lateral Stick Position
Longitudinal Accel (AFT RH Cabin)
Longitudinal Stick POS HI
Pedal Force
Pedal SW Pilot, Co-Pilot
Pitch Bias Actuator Command
Pitch Bias Actuator Position (Feedback)
Pitch Bias Actuator Turn On
Pitch Rate Gyro No 1,2 Output (Unfiltered)
Pitch Servo (SAS 1) No 1 HI, LO
Pitch Servo (SAS 2) No 2 HI, LO
Pitch Trim Command HI, LO
Pitch Trim Position HI (F/B)
Roll Command (SAS 2) No 2 HI, LO
Roll Rate Gyro No 1, 2 Output (Unfiltered)
Roll Servo (SAS) No 1 HI, LO
Roll Trim Command HI, LO
Roll Trim Position (Feedback)
Vertical Accel
Weight on Wheels
Yaw Pedal Switch Supply
Yaw Rate Gyro No 1, 2 Output (Unfiltered)
Yaw Servo (SAS 1) No 1 HI, LO
Yaw Servo (SAS 2) No 2 HI, LO
Yaw Trim Drive Command
Yaw Trim Engage A
Yaw Trim Position (Feedback)
Yaw Trim Turn On 28VDC
Roll/Yaw Artificial Ground
Lateral, Longitudinal, Normal Acceleration
Pitch, Roll Attitude
Pitch, Roll, Yaw Rate
Track (Magnetic Heading)
Airspeed
Angle of SideSlip
AOA
Dynamic Pressure
Pressure Altitude
Total Air Temperature
Wind Direction – Azimuth
Wind Speed
Rate of Climb
Low Airspeed
Ground Speed
Ground Track – True
GPS Latitude, Longitude, Altitude
DGPS Solution Status
GPS seconds into the week
X, Y Position
X-axis North Vehicle Velocity Component
Y-axis EAST Vehicle Velocity Component
Z-axis DOWN Vehicle Velocity Component
Beep Trim Switch (Increase, Decrease)
Engine 1, 2 - Oil Pressure, Temperature
Engine 1, 2 - TGT
Engine 1, 2 Torque
Engine 1, 2 Fuel Flow
Engine 1, 2 Gas Generator RPM
Engine 1, 2 Power Turbine Speeds
Engine 1, 2 Throttles Power Control Levers
Fuel Quantity 1, 2
Main Rotor RPM Low, High
Pneumatic Starter Pressure, Temperature
Main Gearbox Oil Pressure, Temperature
Collective Force, Position
Collective Trim Actuator Position
Lateral Cyclic Force, Position
Longitudinal Cyclic Force, Position
Pedal Force, Position
Pitch, Roll, Yaw SAS Actuator Position
Stabilator Position
Yaw Trim Actuator Position
Tail Rotor Servo Pitch Change Shaft Position
Main Rotor Swash Plate Actuator Position Aft Inp
Main Rotor Swash Plate Actuator Position Fwd Inp
Main Rotor Swash Plate Actuator Position Lateral
Main Rotor Brake Handle Position
Collective Inner Loop Actuator Position
Brake Pedal Deflection Left, Right
Brake Pressure Left, Right
Oleo Deflection Left, Right, Tail
Pilot Event
Pilots Seat Lateral, Longitudinal, Vertical Accel
Tail Wheel Castor Angle
Flight Test Engineer Event

Table 2 **Flight test distribution of test points**

Test type	Number of test points
Accelerations and Decelerations	114
Adverse/Proverse Yaw	30
Autopilot Mode Tests	63
Autorotation	102
Autorotational Entry	8
Azimuthal Winds	348
Climb Performance	282
Collective Control Response	62
Collective Step Inputs	98
Control Authority on Ground	8
Control Forces	11
Control Frequency Sweeps	8
Descent Performance	165
Directional Control Response	207
Directional Static Stability	523
Dive Performance	26
Dynamic Control Releases	40
Engine Tests	110
Hover Performance	41
Hover Turns	5
Lateral Control Response	184
Lateral Directional Oscillations	76
Level Flight Trims	175
Long Term Response	38
Longitudinal Control Response	190
Longitudinal Static Stability	425
Low Speed Level Flight	357
Maneuvering Stability and Steady Turns	339
Minimum Radius Turn	22
Quick Stops	8
Rate of Turn on Ground	55
Roll Reversal Response	155
Short Term Response	149
Sound Recordings	199
Spiral Stability	66
Stabilator Sweeps	473
Takeoffs and Landings	26
Taxi	6
Total (includes 920 test points in ESSS configuration but does include on ground controls, SAS and Autopilot checks.	5194

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37295

AIAA 2000-4584

APPLICATION OF LARGE-ANGLE DATA FOR FLIGHT SIMULATION

Edward G. Dickes*, John N. Ralston†
Bihrle Applied Research, Inc.
18 Research Drive, Hampton, Virginia 23666
Kenneth P. Lawson‡
Naval Air Systems Command
Patuxent River, Maryland 20670

Abstract

Large-angle static and dynamic test data were measured for a scale model of the U.S. Navy F/A-18E/F. Large-angle-of-attack tests were conducted from +90° to +180°and from -90° to -180°. Large-angle-of-sideslip tests were conducted through 90° from -90° to +90° angle of attack. The nominal configuration for these tests consisted of E-model, fighter escort loading (tip and fuselage missiles only), with 34° leading-edge flaps, 4° trailing-edge flaps and +2°/+2° ailerons, with the LEX Vent closed.

Aeromodel increments were generated from the extreme angle of attack and sideslip wind tunnel data for the Boeing/Navy F-18E/F aerodynamic simulation database. In addition, a mechanization structure was provided in the form of additional tables to extend the baseline simulation model coverage.

The application of this large-angle database for flight simulation was successfully demonstrated by the significant improvement in flight correlation and feasibility for the correct modeling of this flight regime for this and other future aircraft.

Introduction

The U.S. Navy, in support of F/A-18E/F Super Hornet EMD flight testing, has sought to continually refine and update their simulation database to provide the highest fidelity engineering simulation of the configuration. As part of the review of the flight motions versus the coverage of the existing simulation database, it was observed that the post-stall motions of this airplane frequently exceed the table limits of the simulation. The ability of the simulation to properly model the subsequent motions and recovery to normal flight is impacted by how the simulation handles these off-table excursions. Further, the Naval Air Systems Command has expressed an interest in improving the understanding of the high angle-of-attack static and dynamic characteristics of their aircraft configurations. To that end, the testing of the F/A-18E/F at extreme angles of attack and sideslip, for static and dynamic test conditions, was conducted to collect the data necessary to extend the database coverage of the baseline simulation, as well as to provide insight to the resultant aircraft characteristics at these conditions.

Large-angle static and dynamic test data were measured for the U.S. Navy F/A-18E/F aircraft using a 1/10-scale model mounted on the rotary balance rig in the NASA Langley Research Center 20-Foot Vertical Wind Tunnel (Reference 1), modified to allow all possible angle of attack and sideslip combinations (e.g., figure 1). Pertinent aerodynamic effects were extracted from these data and assembled into table form aeromodel increments for evaluation in the Boeing/Navy F/A-18E/F simulation as configured from the February 1998 data release. While more recent aeromodel releases have occurred, it is understood that though the most current database implements a different table mechanization, fundamentally the same task is accomplished.

Nomenclature

b	wing span, ft
$\bar{c}$	mean aerodynamic chord, ft
C_A	axial-force coefficient, Axial force/ $\bar{q}$ S
C_N	normal-force coefficient, Normal force/ $\bar{q}$ S
C_Y	side-force coefficient, Side force/ $\bar{q}$ S
C_l	rolling-moment coefficient, Rolling moment/ $\bar{q}$ Sb
C_m	pitching-moment coefficient, Pitching moment/ $\bar{q}$ S $\bar{c}$
C_n	yawing-moment coefficient, Yawing moment/ $\bar{q}$ Sb
p, q, r	roll, pitch, and yaw body-axis rates, deg/sec
$\bar{q}$	free-stream dynamic pressure, lb/ft^2
S	reference area, ft^2
V	free-stream velocity, ft/sec

---

* Senior Aerospace Engineer, Senior Member AIAA
† Engineering Manager
‡ Aerospace Engineer, Member AIAA

This paper is declared a work of the U.S. Government, and is not subject to copyright protection in the United States.

α	angle of attack, deg
β	angle of sideslip, deg
Ω	rotation rate, rad/sec
$\Omega b/2V$	nondimensionalized wind-axis rotation rate, positive for clockwise spin
δ_a	aileron deflection, ($\delta_{a\,left}$ - $\delta_{a\,right}$), deg
δ_h	horizontal tail deflection, + TED, deg
δ_{lef}	leading-edge flap deflection, + LED, deg
δ_{tef}	trailing-edge flap deflection, + TED, deg
δ_r	rudder deflection, + TEL, deg

Discussion of Data

The following discussion reviews selected results from the extensive test matrix. This test program was devised in order to augment and extend test data collected earlier in the program, and as a result, coverage of complete angle of attack and sideslip conditions was based on earlier data availability as well as test rig limitations.

Longitudinal Characteristics

The effect of large angles of attack on the normal force and pitching moment are presented in figures 2 and 3 respectively with neutral and full horizontal tail deflections. As seen in the figure 2, the vehicle's normal force characteristics exhibit a relatively constant coefficient value following both upright and inverted stall through approximately ±120° angle of attack, but decreases rapidly at angles of attack beyond. The pitch characteristics maintain the stable slope seen at 90° angle of attack up through the same ±120° angle of attack point, but beyond these conditions the stability goes from neutral to highly unstable above 150° angle of attack. These data would suggest that for this configuration, the extrapolation of the longitudinal characteristics from 90° to 120° would be a reasonable approximation of these data.

The effect of the horizontal deflection shows that for the most part, the effect of control deflection is constrained to below 90° angle of attack. Additional pitch control (with a slight normal effectiveness as well) is derived as the horizontal tails regain limited effectiveness as they develop lift as a forward control surface at approximately ±145° angle of attack.

The effect varying sideslip to ±90° is shown in figure 4 for selected angles of attack. As seen in these data, sideslip variation has a pronounced non-linear effect on the pitch characteristics for all the angles of attack shown. At low angles of attack, sideslip has a limited nose up effect until very high sideslip conditions are reached. Even at 30° angle of attack, the effect of sideslip remains nose down until sideslip is greater than 45°. As angle of attack is increased further, the magnitude of the sideslip angle at which the transition from nose up to nose down pitch decreases, such that by 70° angle of attack, the configuration exhibits nose up pitching moments for any sideslip condition. As seen in these plots, extreme sideslip excursions can result in significant pitch effects.

The variation of pitching moment with sideslip and rotation for large angles of attack is presented in figure 5. As shown, the pitching moment characteristics at zero sideslip do not vary significantly with rotation throughout most of the high angle-of-attack range, similar to those characteristics seen over the 0° to 90° angle-of-attack range. In the 135° to 145° angle-of-attack region, the pitching moment values become more nose down at the higher rotation rates. The pitching moment vs. $\Omega b/2V$ curves are basically symmetrical for clockwise and counter-clockwise rotations at 0° sideslip angle. At these high angles of attack, the basic rotational characteristics at 0° sideslip angle are essentially unaltered at low sideslip angles. Larger sideslip angles generally skew these curves, resulting in a nose-down increment when the sign of the sideslip angle and rotation rate are the same and, conversely, resulting in a nose-up increment when the signs of sideslip and rotation rate are opposite. The most pronounced effects occur at the highest sideslip angle tested. These effects of sideslip angle on the rotational pitching moment characteristics are typical to those observed over the 0° to 90° angle-of-attack range for the F/A-18E/F and other aircraft configurations, although the lower sideslip angles provide more of an influence on the rotational characteristics of the F/A-18E/F within the 0° to 90° angle-of-attack range.

Lateral Characteristics

The effect of sideslip on the static roll characteristics is presented in figure 6 for selected angles of attack. As seen in figure 6, the effect of sideslip at extreme sideslip angles can have a dramatic effect on the basic stability characteristics of the airplane. At very low angles of attack, little lateral stability is evidenced at any sideslip angle. As angle of attack is increased to near stall (30°), however, lateral stability increases through 45°of sideslip, followed by a gradual transition to lateral instability. By 45° angle of attack the configuration remains very stable with a linear variation with sideslip that abruptly changes to similar levels of instability at 45° of sideslip. The sideslip at which this transition occurs increases as angle of attack increases, up to 60° sideslip by 70° angle of attack.

The effect of aileron through an angle of attack range of ±180° is presented in figure 7 for rolling moment. Limited roll authority persists through ±90° angle of attack, with an abrupt reversal of control power at approximately ±160° angle of attack.

The rotational rolling moment characteristics at high angles of attack as a function of sideslip are presented in figure 8. At 90° angle of attack, the rolling moments at 0° sideslip angle are slightly propelling, a characteristic typical of most aircraft. This propelling tendency gradually increases with increasing angle of attack through 125° angle of attack, with sideslip exhibiting little influence on the rotational variation. By 165° angle of attack, rolling moments are once again, only slightly propelling at the higher rotation rates, with more pronounced sideslip effects on the rotational characteristics. At the extreme angles of attack above 165°, rotation about the velocity vector produces large unstable body-axis rolling moments, and an increase in the effect of sideslip on damping characteristics.

Directional Characteristics

The effect of sideslip on the yawing moment is shown in figure 9. The influence of sideslip is particularly non-linear in the pre-stall angles of attack for this airplane. At low angles of attack the airplane initially exhibits directionally stability that reverses to instability by 20° of sideslip. This unstable trend reverses again at very high sideslip angles. As the angle of attack increases, the region of stability decreases, with substantial unstable directional characteristics persisting to high sideslip. The angle of sideslip at which the directional characteristics reverse to stable slopes decreases as angle of attack increases. This is particularly evident at 40° angle of attack, where significant directional instability is evident through 30° of sideslip, followed by recovery of stable directional slopes. The variation of these transition points illustrates the difficulty in building an appropriate off table approximation (held or extrapolated) for tables fixed at a particular lower sideslip value.

Figure 10 illustrates the variation of rudder control power versus ±180° angle of attack. While rudder power is significantly diminished for upright angles of attack by 60° until achieving limited power when facing into the flow at very high angles of attack, inverted rudder power persists past -90° angle of attack. Also note the presence of forebody induced asymmetric yaw characteristics at both ± 55° to 60° angle of attack at zero sideslip.

The influence of sideslip angle and rotation on the high angle-of-attack yawing moment characteristics is presented in figure 11. The basic aircraft at zero sideslip remains damped in yaw throughout the 90° to 180° angle-of-attack region. A gradual reduction in yaw damping is observed as the angle of attack increases. In general, sideslip has no effect on the rotational characteristics in yaw until high sideslip angles are achieved.

Data Mechanization and Simulation Results

An evaluation of the impact of the large angle wind tunnel test data on the ability of the baseline simulation to model the aircraft behavior at extreme angles of attack was a major component of this effort. Consequently, the presentation of the test data into suitable simulation model tables was performed. These tables were then mechanized in a baseline engineering simulation to enable simulation review and comparison with existing flight test data that exhibited excursions into this flight regime.

The integration of the new wind tunnel data with the existing simulation presented several challenges. The existing model structure inherent in the F/A-18E/F baseline simulation adds the high angle of attack and sideslip data as an increment off of the low angle of attack model (α<40°, β<30°). Consequently, the development of additional incremental effects would have added significantly to an already complex model build up. In order to minimize the work required to add the substantial large-angle data to the existing model, these database extensions were included in the model structure as stand alone tables, with the full coefficient effect inherent in the tables. This modeling required the alignment of the individual coefficients for the baseline configuration, as well as the incremental effects of controls and rotational data for smooth transitioning. In order to arrive at a comparable simulation value to align the new model data with, the simulation was "driven" with state values at the same conditions and test points as the wind tunnel data. The resulting 'virtual wind-tunnel' output was collected into comparable tables that could be evaluated against the new wind tunnel data.

The plotted data were organized as a comparison of the high sideslip (45°) overlap comparisons for the basic airplane, as well as the effects of horizontal tail and rudder, followed by the angle of attack overlap comparisons at ±90°. In general, these results showed good agreement between the simulation and the new test data at these conditions, and the adjustment of the new tables to align with the existing simulation results required only small changes to the new tables to arrive at the desired overlap in most cases. There were other conditions that

required more significant changes to the test data so that the desired identical data values at the table endpoints were achieved. In these cases, the changes were propagated further into the new data set. In the case of the high sideslip tables, initially the 45° table values in the new tables were simply replaced with the current simulation output at that sideslip condition. It was felt that in this manner, the seamless transition from the baseline simulation to the new data could be accomplished without adding additional incremental effects, and without using a ramping function.

The new model data were constructed as separate data regimes, or cases, that enabled the complete coverage of the angle of attack and sideslip space. Based on the value of angle of attack and sideslip, the model transitioned between the cases as shown in figure 12. This modeling called for the substitution of the appropriate tables in the baseline simulation data reconstruction with the modified tables whenever the particular cases occurred. Effectively, as the model excursion went above 45° sideslip, the data that defined the basic airplane characteristics as well as the rudder and horizontal tail effects (the only terms collected during the recent tests in this portion of the envelope), would be replaced in the baseline coefficient build-up. This meant that initially all other terms in the build-up would be propagated into the large-angle conditions.

This new model structure was incorporated into the BAR simulation of the F-18E/F using the FEB98 aerodynamic data set and the Version 7.02 flight control (Reference 2). A modification of the dynamic data mechanization method proposed by Kalviste (Reference 3) was used to mechanize the rotary and body axis dynamic data. Several departure and spin flights from recent flight tests that exhibited excursions into extreme angles of attack and sideslip were selected as examples with which to evaluate the effect of the extended model. The flight state variables from these flights were imported into the simulation and used to "OverDrive" the simulation. A flow chart describing the "OverDrive" process is shown in figure 13. This analog matching technique drives the simulation in a stepwise form using the flight states as inputs to the simulation tables to derive total simulation force and moment output time histories. The resulting simulation moment time histories were compared to those extracted from the flight test data. Several issues became apparent as these comparisons were generated using the baseline model build up. While the incorporation of the new large-angle data had a significant beneficial effect when compared to flight as the airplane excursions went into the newly modeled regions, terms that were carried from the baseline model build-up appeared to have an adverse influence on the coefficient comparisons. This was particularly true in the high sideslip excursion case (Case 1 in the diagram shown in figure 12). Further inspection showed that many of these terms in the baseline model were not modeled as a function of sideslip and the persistence of their influence into the high sideslip region was deemed inappropriate for the desired new mechanization. As a result, the model cases described above were modified to omit all but the new test data as the model transitioned between the Case 0 (baseline model envelope) and Case 1 (high sideslip condition). By transitioning to strictly the high sideslip test data in this fashion, the model correlation in this part of the envelope was significantly improved for the flight test points examined, as shown in the flight comparisons of figures 14 through 16.

Examination of the flights used in this evaluation, which included two departures and an upright spin that transitioned inverted, shows a significant improvement over the baseline model which held table values for off table excursions. While not evaluated in this effort, the effect of extrapolating the off table excursions is expected to yield comparisons at least as dramatic as those shown here, given the wildly fluctuating states that occur as the table look-ups exceed the original table ranges. Another conclusion that can be drawn from these results is that the angle of attack and sideslip are the principal data functionalities at these large-angle conditions, with most other dependencies small in comparison to the large forces and moments generated by these terms. This is evident in the close match between flight and the modified model during excursions into the Case 1 model region, using a database that is primarily a function of angle of attack and sideslip. Because of this, an evaluation of how the baseline model handles some of the other coefficient build-up terms as these extreme angles are approached is recommended.

Concluding Remarks

Large-angle static and dynamic aerodynamic characteristics were measured for a 1/10-scale model of the U.S. Navy F/A-18E/F utilizing the rotary balance test rig in the NASA Langley Research Center 20-Foot Vertical Wind Tunnel. Aeromodel increments of these data have been generated with the extreme angle of attack and sideslip data for the F/A-18E/F aerodynamic simulation database, and were built as additional simulation tables to extend the baseline simulation model's coverage.

The baseline F/A-18E/F simulation model has proven to be a highly effective tool for predicting

flight response through the majority of the airplane's flight regime. Nevertheless, certain departed conditions exhibit extreme variations in angle of attack and sideslip, well beyond the limits of the simulation database. The integration of the newly devised data tables enabled the evaluation of this low speed test data at these conditions. Comparison of the flight extracted coefficient data with results predicted from the baseline simulation and the modified large-angle database showed significant improvement over the baseline model, which held table values for off-table excursions. It was also seen that angle of attack and sideslip are the principal data functionalities at these large-angle conditions, with most other dependencies small in comparison to the large forces and moments generated by these terms. The significant improvement in the flight correlation realized by including a low-speed static wind tunnel database model of this portion of the envelope illustrates the benefit of modeling the fundamental aerodynamics in this flight regime. Further application of these data for future configurations can play an important role in the a priori identification and suppression of large angle departure motions.

References

1. Dickes, E., Ralston, J. and Kloc, S., "F-18E Large-Angle Testing: Static and Dynamic Characteristics From Rotary Balance Wind Tunnel Test Data and Aeromodel Increments", BAR 99-05, June 1999.
2. Kay, J.,"F-18E/F FEB98AERO Model Rehost Static Check Case Verification Results", BAR 99-07M, July 1999.
3. Kalviste, Juri: Use of Rotary Balance And Forced Oscillation Test Data In A Six Degrees Of Freedom Simulation, AIAA Paper 82-1364, August 1982.

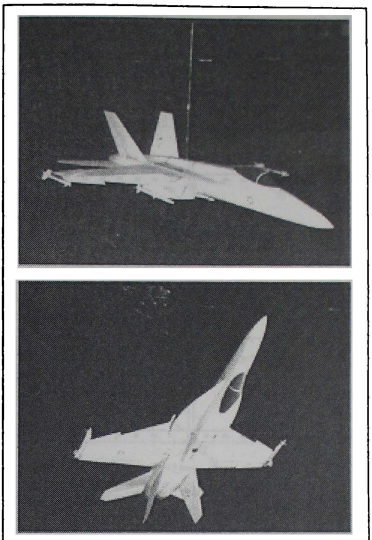


Figure 1. Large-Angle Testing of the F/A-18E/F Model.

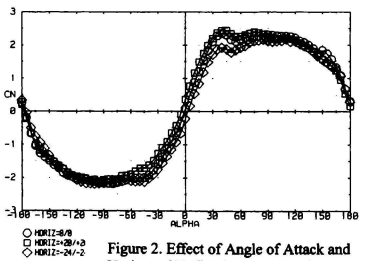


Figure 2. Effect of Angle of Attack and Horizontal Deflection on Normal Force

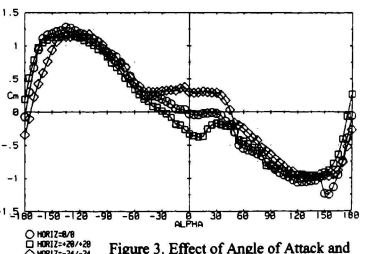


Figure 3. Effect of Angle of Attack and Horizontal Deflection on Pitching Moment

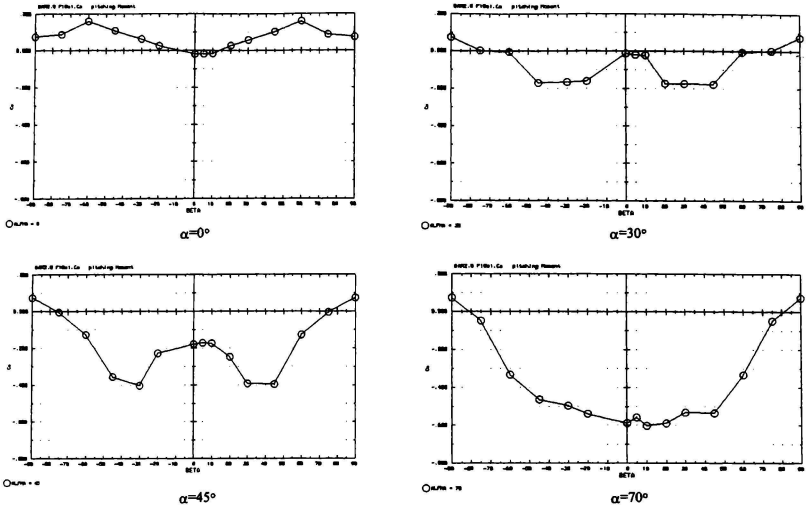


Figure 4. Effect of Sideslip on Pitching Moment

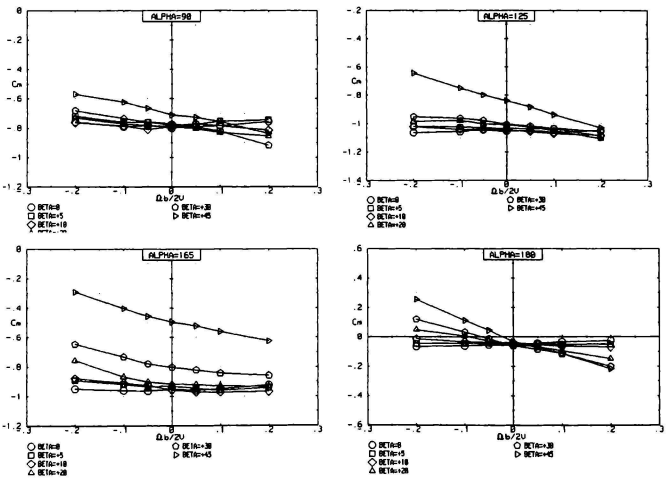


Figure 5. Large-Angle Rotational Pitching-Moment Characteristics

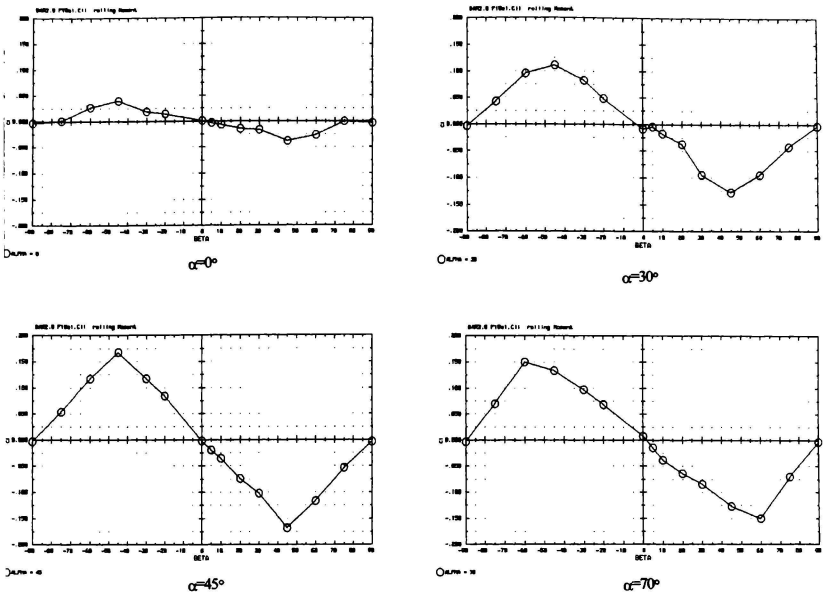


Figure 6. Effect of Sideslip on Rolling Moment

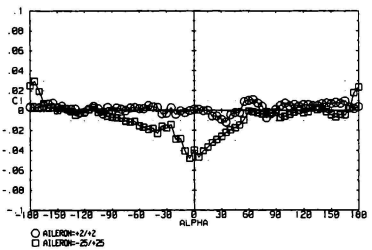


Figure 7. Effect of Aileron Deflection at Large Angles of Attack

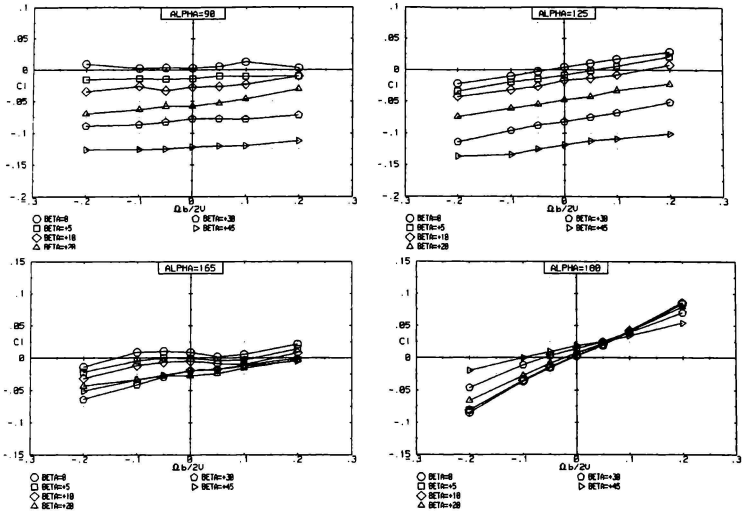


Figure 8. Large-Angle Rotational Rolling-Moment Characteristics

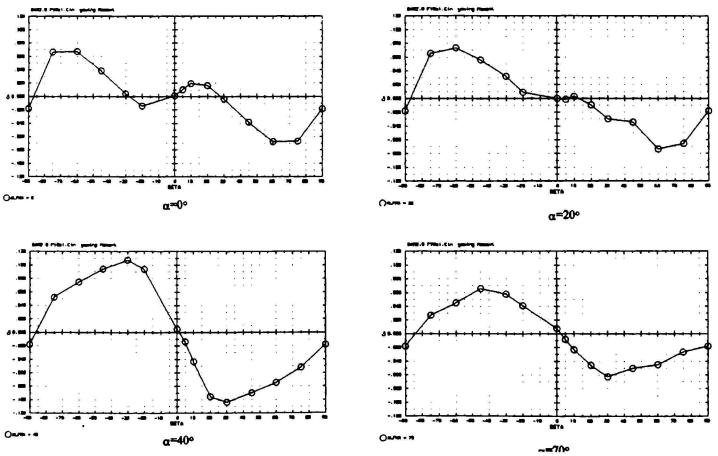


Figure 9. Effect of Sideslip on Yawing Moment

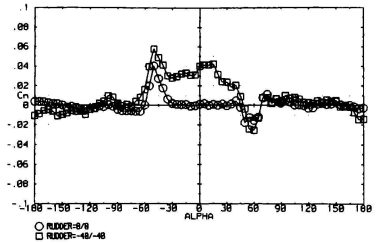


Figure 10. Effect of Rudder Deflection at Large Angles of Attack

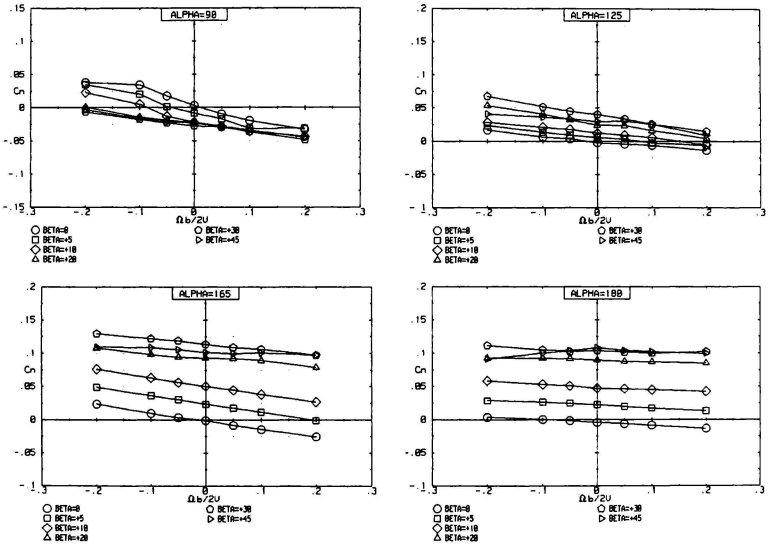


Figure 11. Large-Angle Rotational Yawing-Moment Characteristics

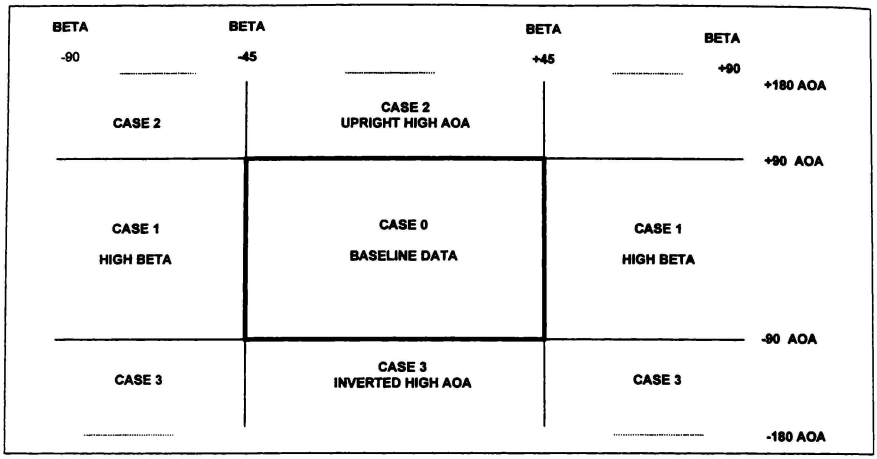


Figure 12. Case Conditions Describing the New Database Extents

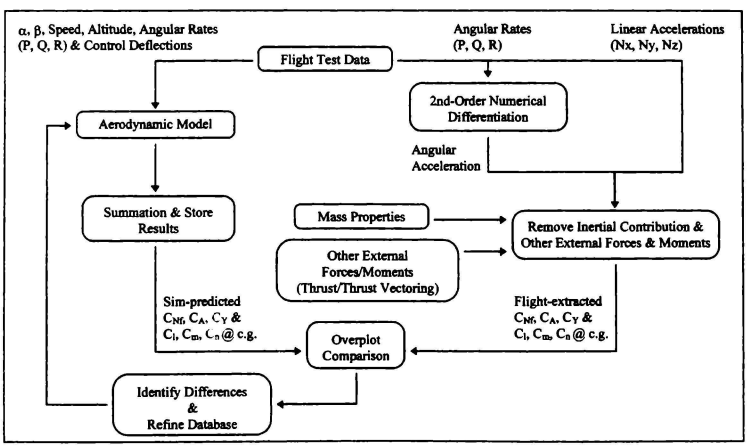


Figure 13. Schematic Of Typical 'OverDrive Technique.

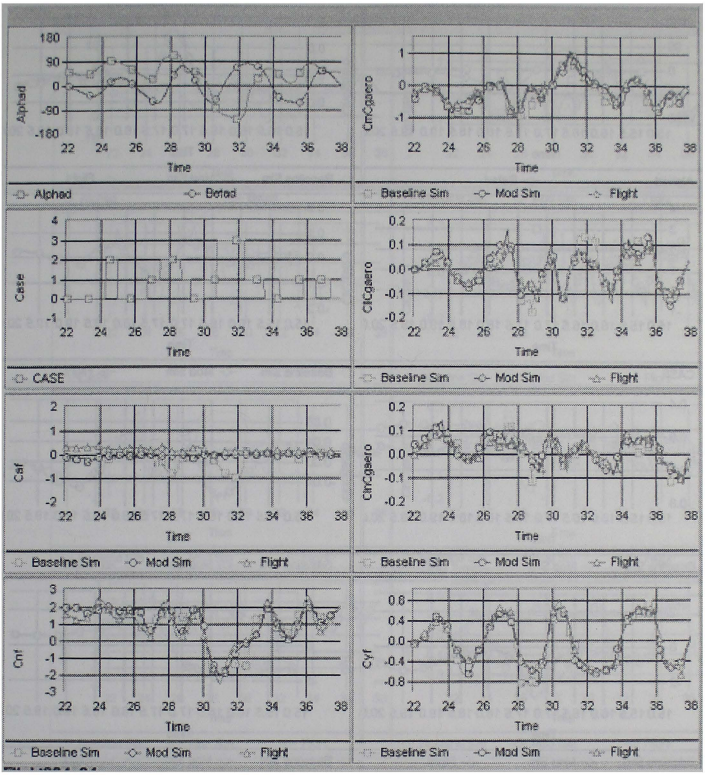


Figure 14.- Departure
Comparison Of Flight Extracted Coefficient Data With Results Predicted From The Baseline Simulation And Revised Database

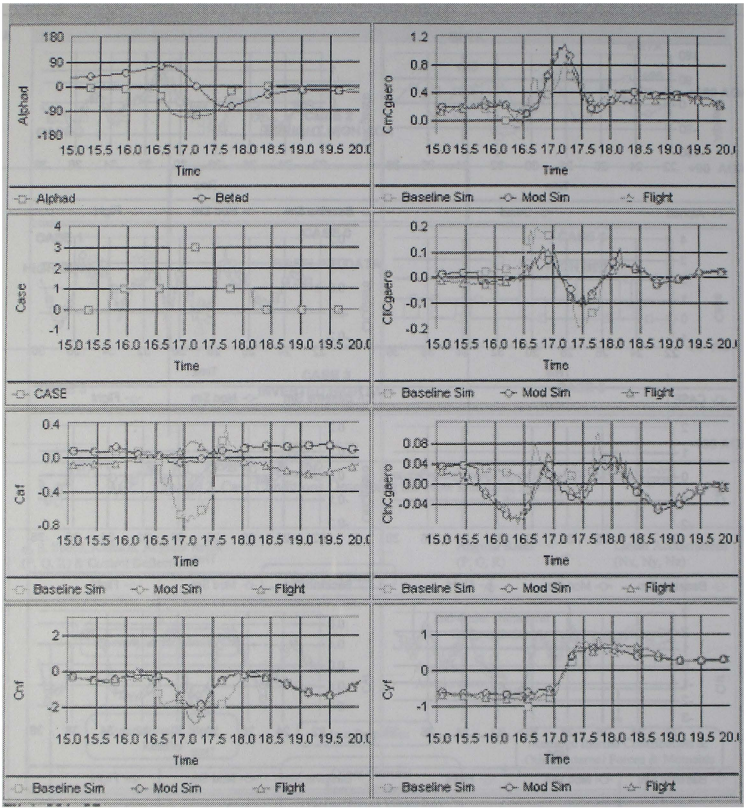


Figure 15.- Departure
Longitudinal Comparison Of Flight Extracted Coefficient Data With Results Predicted From The Baseline Simulation And Revised Database

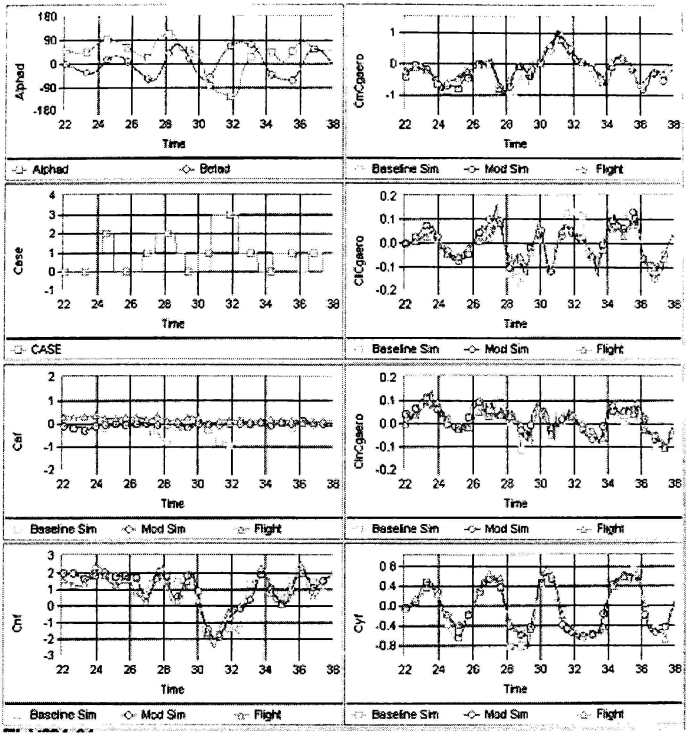


Figure 16.- Upright Spin
Comparison Of Flight Extracted Coefficient Data With Results Predicted From The Baseline Simulation And Revised Database

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37296

AIAA-2000-4585

THE ELEVEN STEPS IN VALIDATION, VERIFICATION, AND ACCREDITATION (VV&A)

Dr. Karl Stengel, System Engineer, nonmember of AIAA
Consolidated Research and Development Contract (CRDC), Joint National Test Facility
The Boeing Company
Schriever AFB, CO 80912-7300
719-567-0893 FAX 719-567-8840 karl.stengel@jntf.osd.mil

ABSTRACT

Verification answers the question "did I build the thing right?" while validation answers "did I build the right thing?" V&V must be done for the conceptual model, the design, the implementation, and the application. (See the Defense Modeling and Simulation Organization Recommended Practices Guide.) This, in turn, requires access to source code, data used in the simulation, and system and requirements documentation. The latter include program plans, technical requirement documents, functional description manuals, user manuals, and in-line documentation. It also requires access to process documentation. This include the Software Development Plan, the Software Configuration Management Plan, the Software Quality Program Plan, and the Work Breakdown Structure or WBS. The VV& agent must also have access to the developers and potential users of the software. This paper discusses how these requirements allow the VV&A process to be divided into the following eleven steps, and examines the steps in detail:

1. Examine Process Documentation
2. Examine System Documentation
3. Interview Developers
4. Interview Users
5. Examine Code 1 (Look For Problems)
6. Examine Code 2 (Break The Code)
7. Determine Truth
8. Generate Test Cases
9. Run Test Cases
10. Review Test Case Output
11. Assessment

Copyright © 2000 by Karl Stengel. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission

Introduction

The major difference between VV&A and IV&V (Independent Validation and Verification) is that the former usually occurs in parallel with the simulation development; the latter afterwards. With the former, the developer can fix any problems in meeting requirements or realism; with the latter it can't. The principles in this paper apply to both, however. According to the DMSO RPG[1],V&V costs vary from 5% to 17% percent of the development cost, with most efforts near 10% to 12%. It will become clear in the rest of this paper that doing VV&A (in parallel with development) is much more efficient (and therefore less costly) than doing IV&V (after the fact). So IV&V costs tend toward the 17% value. VV&A, done in parallel with development, should be close to the 5%-10% figure.

In an ideal situation, the development group tries to anticipate the kinds of tests the VV&A agent might consider. They design and test accordingly. The VV&A agent then does its best to stress the code to its maximum. There is a friendly rivalry, with each group trying to outdo the other. (This applies only if VV&A is done in parallel with development. And then, often, the agent need only watch the Software Quality Assurance or SQA group and the Testing group, understand what they do, and make any needed changes or additions. There is not necessarily any need to repeat the work of those groups.) In this case there is not only cooperation between the developers and VV&A agent, but much less cost because of the lack of duplication.

In a less ideal situation, the developers take it personally if the agent (or SQA, or the Test group) even tries very hard to stress the code. In an R&D environment, code is often prototyped in order to demonstrate a proof of principle. Time-to-market and budget may be critical constraints. This is in contrast with an operational environment, in which code may be used to make life-or-death decisions. Because of these differences, operational code has to be put through extremely demanding tests to ensure it will work for its intended use, hence it requires much more stringent testing than R&D code. If R&D-oriented developers suddenly have to produce an operational code, they will almost certainly be disappointed by the number of tests their code fails. It is not uncommon for a simulation to be in a prototype mode for some time, for an agency to decide that it should be operational, for the simulation to be subjected to a VV&A process when it is really not ready to be operational, and for the simulation to either "fail" VV&A or be severely restricted in its use. The result is the embarrassment of the developers, the disappointment of the agency that decided the simulation was to be operational, and perhaps even the assertion of the funding agency that their money has been wasted. All this pain can be avoided if developers, potential users, and funding agencies are clear on whether a simulation is a prototype or is intended to be operational. Making exaggerated claims or falling prey to excessive marketing pressure is to be avoided.

During verification, the VV&A agent determines whether the simulation meets the specifications that it was designed to meet[2]: "the process of determining that a model implementation accurately represents the developer's conceptual description and specifications." To do this, the agent has to compare the documentation describing those specifications to the simulation. Documentation can be divided into system and process documentation. It also helps if the agent can obtain any specifications or requirements that are not clearly documented. To do this, the agent needs to interview the developers.

During validation, the agent determines whether the simulation meets the needs of the users[3]: "the process of determining the degree to which a model is an accurate representation of the real world *from the perspective of the intended uses of the model.*" To do this, it is logical to interview the users.

Also, to perform both verification and validation, it is necessary to construct test cases, run them, and review them. This review includes comparing the cases with truth, so that determining "truth" (for the purposes of this simulation's uses and this particular VV&A task) is also necessary. Primarily for the purposes of verification, examining the code is also necessary. This can be divided into two tasks – looking for problems, and trying to stress the code to determine the result.

1. Examine Process Documentation

The first of the eleven steps in the VV&A process is to examine process documentation.

This is, among other things, useful in assessing the perceived CMM (Software Engineering Institute Capability Maturity Model) level of the developers. This applies to validation, not verification, because a simulation may do an excellent job of meeting its specifications, but be totally unsuitable for a particular use (the specifications may not match reality, for example).

While it is beyond the scope of this paper to describe the CMM in detail, a short explanation is probably in order. There are five levels[4]. The first is the chaotic, undisciplined process unfortunately characteristic of most software development organizations. The second level covers six key process areas. These are requirements management (ensure you're building what the user wants, and track requirement changes); project planning (have a plan and schedule); project tracking (check progress against schedule); configuration management (as you change requirements or code, keep versions consistent); quality assurance (have an independent group or person ensure you're following your standards and plan); and subcontractor management (make sure your subcontractors follow the standards you set). Most of this is project management 101, the way most companies operate (unless they develop software).

Level Three ensures all aspects of software development (requirements, design, code, test, documentation) follow a standard, internally consistent process which is itself documented. Again, most non-software companies do this. Even custom homebuilders have standard procedures they follow when building a house.

Levels Four and Five are more sophisticated -- they use statistical process control to measure and eliminate defects, and focus on refining processes to make them less bug-prone. Level Four and Five Organizations can even predict approximately how many bugs they will find during testing. Life-critical software usually is (or should be) Level Five.

The U.S. software industry is now going through the same painful process that U.S. manufacturers, particularly the auto industry, went through during the 1970's and the 1980's. They didn't care about quality until they were almost driven out of business by the competition. The CMM is an attempt to build quality into software, rather than spend exorbitant amounts testing and inspecting it in after development. See Port[5] and Stengel[6].

If the developers are at Level One, there is a high probability that the simulation will not meet its specifications, and will contain an unacceptable level of bugs. (One exception is if the simulation is acceptable as a prototype.) If they are at Level Five, it is a virtual certainty the simulation will meet all of its specifications, have very few bugs, and be completely acceptable from a ***verification*** standpoint. (It may still not meet validation standards because it may or may not adequately reflect reality.) From the description of the CMM above, it appears that Level 3 is probably the minimum acceptable CMM level for a non-operational simulation. Level 4 is probably the minimum for an ***operational*** simulation (the simulation is life-critical, or there is a risk of large-scale property destruction if it fails).

There is also a growing tendency in the software industry to self-assess CMM level, and set it by decree. There are cases of development teams being coached to provide the "correct" answers to interview questions, in order to be evaluated as having a higher CMM level than they really do. By following the development process closely, the VV&A agent is in a unique position to more accurately determine the development team's true CMM level.

Alternatively, even if the VV&A agent is not particularly interested in determining the developer's CMM level, examining the process documentation should be useful for evaluations of other simulations produced by that same developer.

The description of Level Two above gives an idea of what to look for in examining process documentation. (Whether or not you wish to evaluate the CMM level, these are clearly things the VV&A agent needs to know.) The first is the process for tracking and controlling requirements. First, is there one? Second, is it being followed? Some of the documents the VV&A agent needs to examine are the Software Development Plan (SDP), the Software Configuration Management Plan (SCMP), the Software Quality Program Plan (SQPP), the Work Breakdown Structure (WBS), and the Test Plans. The SQPP and SCMP help determine whether the developer meets (at least) Level 2 standards in Configuration Management and Quality Assurance, two of the six key process areas of Level 2.

The SDP should contain the following categories: Deliverables, Project Organization, Schedule, Estimates (Cost, Size, Staffing), Risk Management, Software Requirements, Software Design, Coding (including unit test procedures and style standards), Reviews, Integration Testing, Documentation (internal and external), Metrics (planned vs. actuals for size, cost, staffing, milestones, and deliverables), Training (if applicable), Security, and Subcontractor Agreements (if applicable). There should also be a list of Reference Documents.

The SQPP should contain: Reference Documents, Planning, Organization and Responsibilities, Tools, Process Evaluations, Product Evaluations (includes coding standards), Defect Resolution, and Documentation. There should also be checklists for reviews, forms for non-conformance reports, and schedule logs.

The SCMP should contain: Reference Documents, Planning, Organization, Identification (of software and document configuration items), Procedures (for source and document check-in and check-out, and repository backups), Tools, Document Library, Change Control (Process and Tools), SCM Reports, and SCM Metrics.

The WBS should contain categories for Requirements, Design, Code, Test (unit and integration), Reviews, Training (when applicable), Customer Support (when applicable), and Configuration Management. (Generally, SQA should be a separate organization, so there will be no developer WBS category for it.) For a small or medium-sized project, these major categories should suffice. Test Plans should detail integration tests (at least – it's nice if unit tests are included but they seldom are), and their expected results. There should be enough documentation of any test harnesses and inputs to enable the reader to repeat the tests and compare the outputs to the expected results.

While examining the process documentation, the agent needs to determine how clear it is, whether it was actually used the development process, or whether it was developed later. (If VV&A is actually done in parallel with development, the time that the documentation was completed will be clear; during IV&V it may not be so obvious.)

Examining the project estimates and actuals is also useful. Actual time expenditures by WBS category, compared to project estimates, should help provide an indication of any process problems (or, indeed, whether the WBS has enough detail to be used for this purpose). Project metrics include expenditures by WBS category, but also include such things as lines of code developed per person per year, defects per line of code, average time to fix defects, etc. Comparing these metrics to previous projects, and determining how they were derived (of if they were derived) gives more insight into the quality of the development process.

Minutes of walkthroughs or peer reviews are also useful (the latter are usually more formal than the former). This gives an indication of how well the reviews succeeded in finding potential problems, or whether the reviews were even conducted.

Inputs and outputs of developer tests, both unit and integration, give another indication of the quality of the development process. These are useful in constructing test cases, a later step in the VV&A process.

The VV&A agent should also examine the system version description documents. The agent can determine what types of differences existed between various code versions, and how well the documents described them. Finally, the VV&A agent should determine, from the documentation, which DoD or commercial standards were required by the project.

It should be clear that, during this phase, the VV&A agent is performing a type of quality assurance. This clearly can be done more efficiently in parallel with development, with the assistance of the SQA organization. (Recall that the function of SQA is not to evaluate the software ***product***, but the developer ***process***; the VV&A agent is essentially doing the same during this phase.) Working with SQA will also help the agent determine the quality of the SQA process.

2. Examine System Documentation

During verification, the VV&A agent determines whether the software meets functional specifications. Those specifications can be found in such ***System*** documents as the Program Plan, the Technical Requirements Document (TRD), the Operational Requirements Document (ORD), the Functional Description Manual (FDM), etc. In some cases, not all these documents will exist. In others, they will be called by different names. The Program Plan details the schedules and milestones for developing and testing the software. The ORD describes how the software is to be used operationally. The TRD details the functions the software must perform, along with any accuracy requirements. The FDM describes the algorithms, models, and methods used in the software.

The questions this examination should answer: How clear is the documentation? What models and algorithms does the system use? What are the assumptions, or limitations, or constraints, used in the models and algorithms? What are typical inputs and outputs? Are there any types of inputs that can lead to problems? Does the system do any bounds checking or input checking? Does the system produce any warning messages, or simply run? How much user knowledge/training is required? Does the system produce any error messages, or simply crash? Are there differences in use for beginners vs. experienced users? During the testing process, what was output compared to? Does the system use error bounds to increase confidence in the outputs? What is the fidelity of this system perceived to be?

Acceptable documentation (from a ***verification*** standpoint) should answer all of these questions in detail. If the explanations are vague, or nonexistent, the users will probably not have enough information to use the simulation properly. (They may be able to run it, but not understand the outputs – or they may not even be able to run it.) In order to have enough information, the agent will then have to interview the developers.

When using a simulation, users need to know more than just the commands necessary to run it. They need to know how to generate or find the inputs, which inputs are acceptable and which are not, and which inputs are marginal (theoretically acceptable but potentially problematic). They need to understand, at least in general terms, how the simulation derives its answers. (Expert users need to be able to understand that ***in detail*** if they wish to.) They need to know where the simulation can go wrong, when answers are suspect, and what may cause it to crash. They need to know approximately how accurate the answers should be. They need to know whether the simulation will fail gracefully. They need to know how much training they have to have before they can use it. They need to know which cases require minimal training, and which require a lot more.

In the worst case, a user will make an incorrect input, the simulation will run, it will produce an answer that cannot be trusted, and there will be no indication. In a slightly better case, there will be no indication of a problem until the simulation crashes. In a better case still, the simulation will not run but will not tell the user why. Ideally, if an input is wrong, the simulation will not only tell the user, it will tell the user why -- which input or combination of inputs is wrong, and what the acceptable range of values is. Ideally, a simulation will not only produce an answer, but give the user some indication of how accurate the answer is. Ideally, if the simulation is in danger of crashing, it will warn the user.

When VV&A is done in ***parallel*** with development (the preferred situation), any deficiencies in documentation can be caught and corrected during development.

3. Interview Developers

There are several reasons for this step. Often, the documentation (process and system) alone will not answer the questions about algorithms limitations, graceful failure, etc. that were raised in the previous Section. In that case, interviewing the developers is the only way to get this information. Even if the documentation is excellent, interviewing the developers will help determine how closely the documentation was followed. This especially applies to the SDP, any coding standards, the WBS, the SQPP, the SCMP, the metrics, and the test plans. And an outside agency (the VV&A agent) can often bring a perspective that the developers may have missed.

Furthermore, interviewing developers helps determine their CMM Level, and the general quality of the development process, giving an indication of the level of confidence users can have in the simulation's ***verification***. (Recall that a simulation can be exceptionally well ***verified***, but be inadequately ***validated*** if it does not represent reality well.)

The questions that need to be answered are: How closely did you follow (Program Plan, SDP, SCMP, SQPP, test plans, ...)? How did you incorporate requirement changes? Where are they documented? What errors did you look for? How did you look for them? What did you find and fix? How do you categorize, track, and fix problems? Was anything not fixed? What was the basis for the decision not to fix? Which errors are documented? Which are not? Were there changes to documentation or procedures to avoid certain errors? Are there procedures that users need to know which are not documented? How much user knowledge/training is required to use the system? During the testing process, what did you compare to? Do you use error bounds to increase confidence in the outputs? How did you instrument your tests (debug mode etc.)? Which DOD or commercial standards were followed? What is the fidelity of this system perceived to be? How much interaction have you had with users during the development of this system?

Again, it should be clear that interviewing the developers is much easier (sometimes, this is the only way it's even possible) if VV&A is done in parallel with development. And any deficiencies in documentation or process can be found and fixed early, rather than simply being noted after the fact.

4. Interview Users

One purpose of this step is to start validation. The users of the simulation are the ones most likely to know "truth" (as it applies to the simulation). Another purpose is to determine how easy the simulation is to use, by comparing what the users need with what the simulation provides. Even if a simulation represents truth well, it may be unsatisfactory if it's too hard to use.

If satellite orbits are simulated, or data from those orbits is analyzed, the users are probably satellite analysts or astronomers. If the simulation models hit-to-kill kinetic weapons, people working in Theater or Ballistic Missile Defense are probably the users. If the simulation models infrared sensor outputs, sensor designers, TMD or BMD analysts are probably the users. And so on.

Once the users are identified, there are several questions to ask: How do you usually solve these types of problems? What are typical inputs and outputs? What kind of difficulties do you usually have when using this type of simulation? Do you have to be careful in the way any of the inputs are defined? Is some kind of processing required before you have inputs that can be used in this simulation? Do you have to be careful how you use the outputs? Can they be used directly, or is further processing needed? What kinds of inputs lead to unreasonable outputs? What kinds of inputs often lead to failure (simulation crashes, machine freezes, excessive runtime)? How much user knowledge or training is required to understand the inputs, output, and procedures used to solve this problem? To use this simulation? What do you need that other simulations don't provide? Once you obtain an answer from the simulation or procedure, what do you compare it to? Do you use error bounds to increase confidence in the outputs? What is the fidelity of this system perceived to be? What error bounds do you require? What fidelity do you require?

Developers may not talk to the users. If they do, the VV&A agent can work with the developers as they do (assuming VV&A is done in parallel with development). In that case it's much more efficient to do VV&A in parallel with development, rather than afterward. If the developers don't interview the users, it will increase cost for the agent to do it, but it's essential for a credible VV&A effort.

5. Examine Code 1 (Search for Problems)

It is not enough to interview the developers and the users, nor it is enough to examine documentation. Eventually it is necessary to actually examine the code, in detail, to try to find any problems – either with the algorithms used, or the way it meets (or does not meet) specifications. So this step affects both verification and validation.

One of the first things to determine is whether it is possible to input nonsense, or whether the simulation does some kind of input checking. It is relatively easy to check whether inputs fall within a range (between -20 and +55, for example), or whether they are real, integer, or character. Sometimes, however, several inputs which are otherwise acceptable can combine to produce a nonsensical output – cases such as these require either very sophisticated logic in the simulation, or a sophisticated user. If the latter is required, the VV&A agent needs to know.

It's also wise to run the code through a complexity analyzer, such as McCabe[7] or Halstead[8]. Code that appears reasonable may be too error-prone to be usable if it is excessively complex. Since software exists to analyze complexity, it's wise to use it.

It's also important to determine whether the simulation actually uses the algorithms claimed in the documentation. Sometimes code fixes, modifications, or kludges will change an algorithm to something quite different from what was originally intended.

In-line documentation (when it exists) can also help. When it doesn't exist, this tends to indicate hurried, incomplete development. When it does, comparing it with the other documentation can help determine whether algorithms used match those in the specifications. The quality of the in-line documentation – whether it is cursory or detailed, for example – also helps indicate the quality of the software process followed, and the overall (verification) quality of the code.

The VV&A agent should divide the code into modules (if it's not particularly modular, that's an indication in itself), and examine each module for problems. Some of the problems to look for: Divisions by zero; logarithms of zero or negative numbers; underflow or overflow; and exceeding array bounds. Also, even if there is no apparent way for a divisor (or a potential logarithm or square root) to become zero, or nearly so, it is probably wise to have logic to check it anyway. This is particularly true of software which requires external inputs (radar or communications data, for example) – if the input somehow goes awry, zero divisors could occur.

It doesn't really matter whether the simulation has an algorithm problem or a code problem – they both diminish the simulation's credibility (although algorithm problems relate to validation and code problems relate to verification).

Certain languages, more than others, also lend themselves to errors. Languages such as Fortran and Ada usually have compiler switches that automatically initialize all variables to negative indefinite, so that if those variables are subsequently used without being initialized, the simulation will crash. In this way it is virtually assured that all variables are defined. Languages such as C++ do not have these switches. Also, the pointers used in C++ are highly prone to error. This does not make C++ unacceptable; it merely means simulations written in it must be more carefully examined for memory leaks and undefined variables.

If the simulation generates some kind of error bounds, or fidelity estimates, it's important to see how these are calculated, and to try to determine when they might be inaccurate. To do this, it's generally necessary to understand the theory of the bound calculation, and its limitations. It is preferable that the documentation explain this; if not, the users or developers may yield some insight.

It's again more efficient to follow this step in parallel with development, rather than afterward. Any problems can be fixed early, as opposed to perhaps not being fixed at all.

6. Examine Code 2 (Break it)

This step goes beyond looking for problems; here the VV&A agent deliberately tries to cause them. The users need to know when they will get into trouble; the agent needs to tell them. When VV&A is done in parallel with development, this process should closely follow what the developers are doing during testing (and during design and coding – anticipating problems early on saves a lot of time later, during testing). This step is primarily verification.

First, examine the interfaces between modules. Will data transferred ever be missing or incomplete? If so, this can lead to undefined variable errors. Will it ever default to unacceptable values? Also, trace back through the various modules. If you create a failure in one module, can it be traced back to an acceptable input? Trace forward through the modules. If you pass acceptable data into one module, can it cause a problem several modules later? This kind of analysis is usually not susceptible to formal methods. It takes a certain amount of insight, creativity, and familiarity with the code and what it is attempting.

Next, attempt to create stressing cases based on (at least) the number of objects (vehicles, sensors, interceptors, etc.), the number of users, the memory used, the disk space used, and the run time. The latter includes cases deliberately constructed to take unacceptable run times. (In this case, the recommendation may be that, while theoretically the simulation can run these cases, for all practical purposes they should not be run.)

Check boundaries. The latter includes such things as examining loops that run from one to a hundred, and checking the results when a value is zero, one, 99, 100, or 101. It also includes examining (if the simulation has a maximum of fifty interceptors) what happens with 49, 50, or 51.

Examine the algorithms used, and attempt to stress them. What happens if a denominator can be nearly zero? What happens if an approximation is used, and a case runs where the approximation is poor?

The test group should be doing this kind of work during development, and the VV&A agent can (theoretically) simply follow along, making suggestions as appropriate. If VV&A is done after the fact, however, this work can be very time consuming and expensive.

7. Determine Truth

The "truth" in this step is the real world, or the part of it, that the simulation is trying to model. In this step, the VV&A agent performs a lot of the work necessary for validation. Usually, the agent must interview the users first -- they can either define the truth, or tell the agent how it can be determined. Other systems may also provide this information. If the developers are building a real-time code, and a well-accepted code that runs much more slowly already exists, truth can be found from the slower code.

Other sources are wargames, empirical data such as flight tests, and laboratory measurements (when appropriate and available). If a sensor is being simulated, bench tests of the sensor, or flight tests of it, can provide data which can be compared to simulation output. In some cases, calculations done by hand or with a calculator, and using accepted physical principles, can also provide truth. For example, an orbit calculation routine can be tested (in part) by comparing its predictions for circular orbits to hand-calculated results. Rough estimates of sensor output can be derived from blackbody equations, and geometric considerations.

The questions the VV&A agent must answer are: What kind of results are expected, when this code runs on the cases of interest? How do we know what to expect? How certain of the definition of truth are we? If there is really no good definition of truth, it's impossible to do validation, because there's nothing to compare simulation output to.

The requirements should reflect the needs of the user, but if they don't the simulation can still be well-constructed from a verification standpoint, even if it fails from a validation standpoint. If the developers perform this step, the VV&A agent can often follow along, making any needed additions or corrections. In that case it's much more efficient to do this part of VV&A in parallel with development. If the developers don't determine truth, it increases the cost for the agent to do it, but it's necessary for a credible VV&A.

8. Generate Test Cases

This step will generally use the results of all the previous seven steps (with the possible exception of process documentation). System documentation, user interviews, developer interviews, code examination (both kinds), truth determination – all will, potentially, lead to test cases. Then the output from those cases will have to be compared to truth.

What are the acceptability criteria? For a given case, what should the output be? How close should it be to the truth? This has to be defined, specifically, before that case is run – if you don't know how close you need to be, you can't say whether you're close enough once the case is run.

Test cases are simple, typical, and stressful. "Simple" are the types of examples the user manual contains to illustrate the simulation's operation – given these inputs, these are the types of outputs to expect. "Typical" can cover a wide range, depending on what the simulation is used for and what users encounter. This is one time when user interviews, and examination of the system documentation, is particularly valuable. Finally, cases which stress memory, or run time, or complexity, must be run they are to be considered when the simulation is actually used. If the simulation will not be used for anything other than typical cases, whatever those are, there is no need to test for stressful ones.

As-built tests are not full verification tests. They are based on the way the code is written, based on code examination. There also have to be tests to confirm that the requirements are met. But these are not enough. Validation tests are also needed. The simple, typical and stressful cases described above will generally be validation tests, provided the users have helped construct them. This is one area where many developers will not be adequately prepared. If they haven't worked with users, their requirements may not reflect user needs, and the simulation then won't either, no matter how well it's built.

Doing this step in VV&A after the fact will be quite expensive, since the agent will have to develop the test cases from scratch (unless the users, developers, and documentation provide very specific ones).

It is important to state that test cases not only include inputs, but expected outputs, and acceptability criteria (how close does the output need to be to that expected?) Acceptability criteria may also include run time, disk usage, or CPU usage. If we need an answer in twenty minutes but it takes three hours, perhaps we can't use the simulation for that case.

In order to obtain outputs from the test cases (the next step), it may be necessary to build test harnesses, or otherwise instrument the code. The developer should be doing this; this is yet another reason to do VV&A in parallel with development, so as to avoid duplication. Pre-processors (to put inputs in the correct form) and post-processors (to put output in an understandable form, which can be compared to expected outputs) may also have to be built. The developers should do this; the VV&A agent can simply watch, making any necessary additions or changes. If the developers are not doing this, and the agent has to do it (or has to force the developers to), this speaks volumes about the credibility of the code and the development process.

9. Run Test Cases

Once the test cases are created, the agent must run them. If VV&A is done in parallel with development, the agent can watch the developer do this, making any additions that might be necessary. Running test cases after development is complete can be quite expensive.

The inputs to the test cases should have been defined in the previous step. They should be run through any necessary pre-processors. The outputs should be obtained directly, or from instrumented code, or from code run in debug mode (when the languages and compilers used permit). They may need to be run through post-processors before they can be compared to expected results. The need to build test harnesses (instrument the code), pre-processors, and post-processors can particularly run up the cost; that's another reason to do VV&A in parallel with development, since the developers generally build such tools anyway.

The outputs should be in such a form that they can be readily compared to the expected results from the previous step. It should also easy to apply the acceptability criteria (and such criteria should exist, having been defined in the previous step).

10. Review Test Case Output

Compare test case outputs to expected results, and apply the acceptability criteria. Are the requirements met? Do the validation cases produce the expected results? Since the developers should be doing this anyway, as long as VV&A is done in parallel with development, the agent should be able to watch the comparison the developer does (and make some necessary changes and additions). Once again, many developers will fall short here. They will have verification tests, but not validation tests. (Often they will not even have adequate verification tests.)

Generally, there has to be some iteration here. Some of the test case output will not be acceptable, and code changes will be needed (when VV&A is done in parallel with development). If VV&A is done afterward, unacceptable test case output will lead to rejection of the code for at least some of its intended uses. It may result in a warning – "use the code for cases A through J, but not for cases K through Z" – or it may result in outright rejection.

The agent (ideally, in concert with the developer) must identify the causes of any discrepancies. The problem has to be traced back through the code, and the cause identified. Was it user input? An algorithm problem? A coding problem? A hardware problem? Unclear requirements? This has to be done, not only for output that is too far from the truth, but for any freezes or crashes as well. The latter is particularly important for any operational software. If lives are riding on the software, a crash or freeze can be fatal.

The agent must determine whether output is close enough (the spec said ten percent, it's eleven; that's close enough) or not (the spec said ten; it's fifty). The cause of the difference has to be found, and either fixed (code change, hardware fix, requirements rewrite, user input check) or documented (don't use the code for this case).

The agent also must check for repeatability problems (the answer varies depending on which cases were run earlier – a sign of potential memory leaks), excessive memory use (use all the system memory and the simulation will freeze or crash), excessive disk use (which can lead to a freeze or crash), and excessive run time (there's either a problem with the code since it's taking longer than we expect, or we can't use the code for this case since it takes too long).

11. Assessment

This is the final step! It's essentially a summary of the previous ten steps, with a recommendation as to whether the code can be used for its intended purpose. There should be an assessment of the simulation's performance, of the code, of the documentation, and of the user (what the user needs to know to use the simulation).

There should also be an assessment of the perceived CMM level of the developers. Generally, the higher the level, the greater the credibility of the simulation with respect to meeting requirements. Even if it meets requirements (is verified), it may not meet user needs (be validated). This also helps to evaluate the developers, if they produce other simulations in the future. It also points out any particular problems the developers may have in their software process – inadequate testing or documentation, ill-defined requirements, poor design, etc.

For the performance assessment, how close does the output come to truth? How close does it need to be? How well can the truth be determined? How easy is it to compare the output to truth? How thorough are the comparisons (output to truth)? Are there error bounds? Are there cases of crashes, freezes, or failures? If failure occurs, is it graceful? Are there workarounds to problems?

For the code assessment, how complex is the code? How well is it designed? It is modular? Is it object oriented? Is it robust? These issues may not immediately concern the users, but certainly make a difference during maintenance or upgrades. And – does it include examples, or test cases, or post-processors, or pre-processors? This can make a great deal of difference in how easy the code can be used.

For documentation assessment, examine the in-line documentation and the user documentation. How well does it help the user run the simulation? Does it include examples of inputs and outputs? Does it describe the algorithms and assumptions made? Does it describe limitations or problems, and workarounds? Does it describe the simulation as built, or as designed? Is it complete?

For user assessment, does the simulation calculate the things the user wants? How easy is it to use? If the code does a lot of careful input checking, and provides a lot of on-line help (or manual help), the user may be able to use it with little training or

preparation. If improper inputs produce an immediate error message, that's good. If an input error is detected only when the code crashes, or produces an inaccurate output, that's bad. Code that's hard to use may still be usable, but the users need to know about the difficulties. The level of user training or knowledge assumed or expected should be noted. This can be critical – a simulation can be wonderful if the users are well-trained – and unacceptable if they are not. The assessment MUST note this.

Finally, the recommendation: Is the code suitable for its intended use? The answer can be "yes", or "no", or "maybe." The latter can be "for the following cases", or "if the following pre-processors and post-processors are included", or "if the following documentation is provided", or "if the following training is provided", or "if the following changes are made (to documentation, to user training, or to code)", or "more V&V must be done first." In practice, the answer will seldom be (an unqualified) "yes". Most codes have limitations – they can't be used for certain cases, or they require a certain level of user training, or a certain level of user sophistication. The report must also describe the intended use, since that may change with time, or different users may have different uses in mind.

References

1. Defense Modeling and Simulation Organization (DMSO) Recommended Practices Guide (RPG), November 1996, page 1-20 (DMSO VV&A Site is at: http://www.dmso.mil/home/vva;

   DMSO RPG is at ftp://ftp.dmso.mil/pub/documents/initiatives/vva/rpg/rpg_frontmatter.pdf

2. DMSO RPG, page 1-5.

3. DMSO RPG, page 1-7.

4. For detailed descriptions of the CMM, see the Software Engineering Institute (SEI) Web Page: http://www.sei.cmu.edu/

5. "Will Bugs Eat Up the U.S. Lead in Software?" Otis Port, Business Week International Edition, 6 Dec 1999 (on-line at http://www.businessweek.com/1999/99_49/b3658020.htm)

6. "Software Developers Need to Take Work to Next Level", Karl Stengel, Vox Pop, Denver Post Business Section, 8 August 1999

7. Complexity Measure, McCabe, IEEE Transactions on Software Engineering, Volume 2, No 4, pp 308-320, December 1976. McCabe Complexity measure at Software Engineering Institute (SEI) website: http://www.sei.cmu.edu/activities/str/descriptions/cyclomatic_body.html

8. Elements of Software Science, Halstead, New York, Elsevier North-Holland, 1977. Halstead Complexity at SEI website: http://www.sei.cmu.edu/activities/str/descriptions/halstead_body.html

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

A00-37297

AIAA-2000-4586

FLIGHT DATA RECONSTRUCTION AND SIMULATION OF THE 1992 AMSTERDAM BIJLMERMEER AIRPLANE ACCIDENT

M.H. Smaili*
National Aerospace Laboratory NLR
Anthony Fokkerweg 2
1059 CM, Amsterdam, the Netherlands
Tel. +31 20 5113013, Fax +31 20 5113210
e-mail: smaili@nlr.nl

J.A. Mulder†
†Delft University of Technology
Kluyverweg 1
2629 HS, Delft, the Netherlands
Tel. +31 15 2785378, Fax +31 15 2786480
e-mail: J.A.Mulder@lr.tudelft.nl

ABSTRACT

On October 4th 1992, a Boeing 747-200F Freighter Airplane lost its right-wing engines after departing from Amsterdam Schiphol Airport. Due to severe performance and controllability problems caused by this the aircraft crashed, 13 km east of the airport, in the Bijlmermeer, a suburb of Amsterdam. In recent years, several similar incidents have occurred in which aircraft were successfully recovered after encountering a separation of one or more of the engines. This paper describes state-of-the-art computer simulation techniques that were applied during an independent analysis of the accident performed at the Faculty of Aerospace Engineering of the Delft University of Technology in 1997. Utilising high performance computation and visualisation software, a reconstruction of the flight was performed using the parameters of the Digital Flight Data Recorder (DFDR). The reconstruction method, referred to as Flight Data Reconstruction and Simulation (FDRS), allowed an accurate estimation of the flying capabilities of the aircraft after the separation of the engines. The analysis indicated that from a technical point of view the aircraft was recoverable despite the severe performance and controllability problems caused by the separated engines. For future research on advanced avionics/ flight systems design, the reconstructed model resulting from the analysis can be utilized as a benchmark to evaluate flight control concepts on their performance to accommodate in-flight failures.

---

* Master of Science Aeronautical Engineering (MSc), AIAA member

† Professor, Faculty of Aerospace Engineering, AIAA member

Copyright © 2000 by the National Aerospace Laboratory NLR. Published by the American Institute of Aeronautics and Astronautics, Inc. with permission.

NOMENCLATURE

CL	= lift coefficient
CY_β	= side force coefficient due to sideslip
CY_{delta_r}	= side force coefficient due to rudder
ΔCD	= drag coefficient due to wing damage
ΔCL	= lift coefficient due to wing damage
CNe	= yawing moment coefficient due to thrust asymmetry
$c.g.$	= center of gravity
ΔYcg	= Lateral c.g. distance from aircraft centerline
lv	= rudder side force moment arm
lwd	= Lateral distance from aircraft centerline of acting point of lift loss and drag due to wing damage
Tn	= engine thrust
V	= true airspeed
β	= sideslip angle
$delta_r$	= rudder deflection
ϕ	= bank angle

ABBREVIATIONS

ATC	Air Traffic Control
CAD	Computer Assisted Design
DASMAT	Delft University Aircraft Simulation and Analysis Tool
DFDR	Digital Flight Data Recorder
DUT	Delft University of Technology
FDRS	Flight Data Reconstruction and Simulation
EGT	Exhaust Gas Temperature
EPR	Engine Pressure Ratio
GA	Go Around
IAS	Indicated Airspeed
ICAO	International Civil Aviation Organisation
MCT	Maximum Continuous Thrust
NLR	National Aerospace Laboratory
PF	Pilot Flying
TOGA	Take Off/Go Around
UTC	Universal Time Co-ordinated

INTRODUCTION

After departing from Amsterdam Schiphol Airport on October 4th 1992, a Boeing 747-200F Freighter went down near the airport after an encounter of a multiple right-wing engine separation. In an attempt to return to the airport for an emergency landing, the aircraft flew several right-hand circuits in order to lose altitude and to line up with the requested runway. During the second line-up, the crew apparently lost control over the aircraft. As a result, the aircraft crashed, 13 km east of the airport, into an eleven-floor apartment building in the Bijlmermeer, a suburb of Amsterdam. Following the accident, an investigation was initiated by several departments and authorities. The Netherlands Accident Investigation Bureau, charged with the investigation, was assisted by specialists from the Aeronautical Inspection Directorate of the Department of Civil Aviation[1]. According to the procedures contained in International Civil Aviation Organisation (ICAO) Annex 13, accredited representatives and their advisors from several countries joined the investigation. As far as the technical aspects of the flight were concerned, NLR, the National Aerospace Laboratory of the Netherlands, was tasked with several projects. The aircraft manufacturer performed an analysis of the Digital Flight Data Recorder (DFDR) of the flight and examined its results by means of piloted simulations[2]. The results of the investigation, however, were hampered by the fact that the actual extend of structural damage to the right wing was unknown. Although the controllability aspects could be reproduced within reasonable tolerances in the simulator, the performance aspects showed discrepancies. Especially the last minutes of the flight, and the subsequent loss of control, raised questions that were solved relying on the data of the DFDR. The origin of several anomalies in the flight control system, contributing adversely to the control of the aircraft, remained yet unknown. The analysis concluded that given the performance and controllability of the aircraft after the separation of the engines a successful landing was highly improbable[1].

In 1997, the Division of Flight Control and Simulation of the Faculty of Aerospace Engineering of the Delft University of Technology (DUT) in the Netherlands performed an independent analysis of the accident[3]. The analysis applied modeling, simulation and visualisation techniques for a reconstruction of the flight mechanics of the aircraft using the DFDR pilot control inputs. DFDR data for the analysis was obtained from NLR and the Netherlands Aviation Safety Board[4]. The purpose of the analysis was to acquire an accurate estimate of the actual flying capabilities of the aircraft and to study alternative flight scenarios for a successful recovery. This paper describes the computer simulation techniques applied during the accident analysis and gives a summary of the obtained results.

OVERVIEW OF THE ANALYSIS

The Flight Data Reconstruction and Simulation (FDRS) method (ref. [12]), applied the DFDR pilot control inputs to a detailed simulation model of the aircraft. The sampled DFDR pilot control inputs were applied in an inverse simulation method to obtain a best match of DFDR measurements and simulation. In this approach, the simulation model virtually 'flies' the accident profile, according to the pilot's control inputs, thereby reconstructing missing flight data and any fault events that lead to the loss of the aircraft. The simulation environment, developed for the analysis, enabled to assess the flight mechanics and control effects by means of visualisation. Using the reconstructed model, failure mode and effect analysis was applied to the flight control system. The reconstruction method proved to be a practical tool for estimating the aerodynamic and overall flight mechanics effects of engine separation. Visualisation facilitated comparison with the DFDR data. The actual flying capabilities of the impaired aircraft were next investigated by applying alternative control strategies to the reconstructed model.

The reconstruction method applied during the analysis resulted in a simulation model of the impaired aircraft that matched reasonably well with the performance and controllability effects as recorded on the DFDR. The introduction of control loss could be visualised in detail using additional flight mechanical parameters. In this way the applied control inputs during the last flight stage could be analysed in addition to other flight mechanical aspects. Failure mode and effect analysis gave more insight into the performance of the flight control system before and after the separation of the engines. Results of the reconstruction were also used to substantiate additional data on the aircraft's flight path using eyewitness reports[5]. The actual flying capabilities of the aircraft to perform an approach and landing were examined using the reconstructed model.

SEQUENCE OF EVENTS[1]

The accident aircraft was scheduled for a flight to Ben Gurion International Airport, Tel Aviv, with an intermediate stop at Amsterdam Schiphol Airport after a flight from John F. Kennedy International Airport, New York. The aircraft received an air traffic control slot time of 17:20 (UTC) for departure. The

aircraft was refueled with 72 metric tons of Jet A1 fuel and was loaded with a total of 114.7 metric tons of cargo. The takeoff gross weight was determined to be 338.3 metric tons.



Figure 1: The aircraft landing at Amsterdam Schiphol Airport on October 4th 1992 (Studio LCP)

At the time of departure, the preferential runways consisted of runway 01L (Zwanenburgbaan) for take-off and 06 (Kaagbaan) for landing. The aircraft was cleared for push back at 17:04 and taxied out at 17:14. The first officer was assigned as the pilot flying (PF). The takeoff from runway 01L was started at 17:21, and the aircraft was cleared by ATC for the Pampus departure.

At 17:27.30, while climbing through an altitude of about 6,500 feet, the aircraft encountered a separation of the engines no. 3 and 4. The captain took control of the aircraft. Following engine separation, the emergency call "mayday, mayday, mayday, we have an emergency", was transmitted by the co-pilot. The aircraft started a right turn to return to the airport for an emergency landing. According to eyewitnesses, dumping of the onboard fuel started immediately. Amsterdam Radar confirmed the emergency call and directed the flight during the emergency procedure. After the crew acknowledged their intentions, they were instructed to turn to heading 260.



Figure 2: The aircraft returning to the airport after separation of the no. 3 and 4 engines

At 17:28.17, the crew reported a fire on engine no. 3 and they indicated a loss of thrust of both engines 3 and 4. At 17:28.57, the aircraft was informed that the main runway for landing was runway 06. The wind at that time was 040° at 21 knots. The crew of the flight, however, requested the use of runway 27 for landing. Because the aircraft was only 7 miles from the airport at an altitude of 5,000 feet, a straight-in approach would not be possible. ATC instructed the crew to a heading of 360 degrees to fly a circuit and to descend to 2,000 feet. By then the wind was 050° at 22 knots.

At 17:31.17, the crew indicated that they needed "12 miles final for landing". During the transmission of this reply, the crew commenced the selection of flaps 1 for landing. While instructed to turn right to heading 100 the crew reported 'no. 3 and 4 are out and we have problems with the flaps". After the aircraft was established on heading 120, the crew maintained an indicated airspeed of 260 knots and a gradual descent. ATC cleared the flight for approach and instructed a heading of 270 to intercept the final approach course. Indicated airspeed remained at 260 knots at an altitude of 4,000 feet. After the heading instruction from ATC, it took about thirty seconds before the heading change was actually performed. When it became clear that the aircraft was going to overshoot the runway centerline, ATC instructed the flight to turn further to heading 290 to intercept the localizer from the south. Twenty seconds later a new heading of 310 was instructed by ATC, along with the clearance to descent to 1,500 feet.

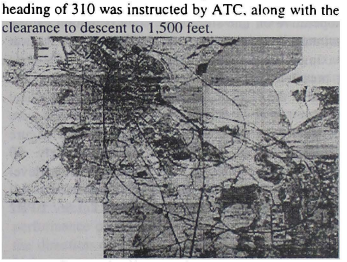


Figure 3: Flight path of the aircraft

At 17:35.03, the crew acknowledged the clearance by reporting "...1500... and we have a controlling problem...". At this point, indicated airspeed decreased to 256 knots. The crew was losing flight control and approximately 25 seconds later the captain called, "going down 1862, going down...". During this transmission, the crew was trying to recover the aircraft by raising the flaps and by lowering the gear. The stick shaker and ground proximity warning system were audible in the background of the transmission. The remaining engines no. 1 and 2 were set at maximum thrust.

At 17:35.42 the aircraft impacted at a roll angle of approximately 104°, a load factor of about 2.5 g's and approximately 70° pitch down.

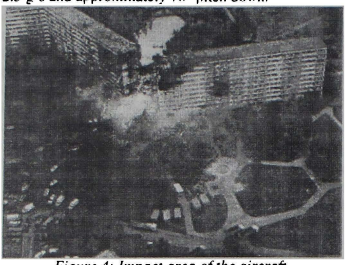


Figure 4: Impact area of the aircraft

ANALYSIS OF THE FLIGHT

Control Capabilities

Under nominal conditions in the case of a failure of both right-wing engines without separation, aircraft should have the capability to turn in either direction with adequate control authority. The accident aircraft was designed to have enough rudder authority to keep the control wheel almost neutral with two engines inoperative on one side. This flight condition can be maintained up to the remaining engines set at maximum continuous thrust (MCT/EPR 1.35) while at maneuvering speed. For the case of the accident aircraft, the DFDR indicates that control wheel deflections between 20 to 60 degrees to the left were needed for lateral control and straight flight (figure 5). The largest deflection of approximately 60 degrees was required for straight and almost level flight. This condition could only be maintained at full rudder pedal and at high thrust (EPR1 1.56/ EPR2 1.45).

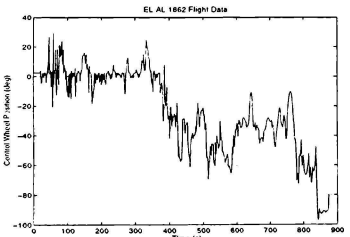


Figure 5: DFDR control wheel deflections

According to the DFDR, maximum available rudder was needed during the straight legs to counteract the yawing moment, the traces of the rudder control surface activity as a response to the rudder pedal inputs can be seen in figure 6. In this figure, a limited control authority of the lower rudder is visible. From the DFDR it can be determined that lagging of the lower rudder started after the first turn, approximately 100 seconds after engine separation (t=378 s) and at full left pedal. Pedal relaxation during turn initiations caused the lower rudder to follow the upper rudder again. At final loss of control and increasing roll angle, the DFDR shows a sudden increase of lower rudder deflection while the upper rudder stays behind at a smaller deflection.

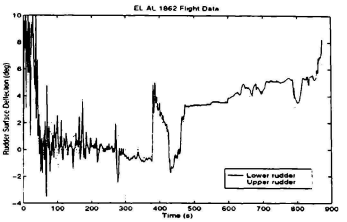


Figure 6: DFDR rudder surface deflection

The DFDR indicates (figure 7) that the controllability and performance condition after engine separation required engine no. 1 and 2 thrust settings between approximately MCT (EPR 1.3) and overboost thrust (EPR 1.62). High thrust (EPR1 1.56/ EPR2 1.45) is needed to sustain almost straight and level flight. This condition is reached approximately 120 seconds after separation of the engines and after completion of the first turn towards a heading north.

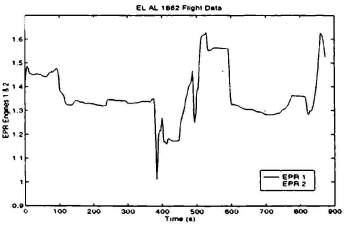


Figure 7: DFDR engine no. 1 and 2 thrust settings (EPR)

In general, the DFDR indicates that both pedal and control wheel were used for turn initiation and roll control. The first turn after the separation of the engines no. 3 and 4 is performed at almost zero pedal deflection. At final control loss, control wheel deflection is maximum while rudder pedal deflection is less than maximum.

An analysis of the flight in a flight simulator[2] indicated that in the above mentioned conditions and with maximum rudder pedal input, approximately 30 degrees left control wheel deflection was needed to maintain straight flight. This condition was simulated with trailing edge flaps up. According to the hydraulic system architecture, this condition locks the outboard ailerons when outboard flaps are not selected. For the case of the accident flight, the additional wing damage and degraded effectiveness of the right-wing inboard aileron required larger left wing down control wheel deflections than in the nominal case. This effect can be determined from the DFDR and was confirmed by reconstruction of the flight.

The above analysis taken into account, it is clear that the crew of the aircraft was confronted with a flight condition that was different from a nominal two engine out situation. For the heavy aircraft configuration at a relative low speed of 260 knots IAS, the DFDR indicates that flight control was almost lost at full pedal, 60 to 70% of maximum lateral control and at high thrust.

Performance Capabilities

An energy analysis of the flight using the DFDR data[6] indicated that after the separation of the engines the aircraft had level flight capability at go-around thrust (GA) and at an airspeed (IAS) of approximately 270 knots. Maneuvering capabilities were marginal and resulted into a loss of altitude. A normal load of approximately 1.1-g, equivalent to 25 degrees of bank, reduced the maximum climb capability to approximately minus 400 feet/min. At MCT thrust and at approximately 270 knots IAS, maximum climb performance was about minus 350 feet/min. Below 260 knots IAS, a normal load of 1.15 g and an angle of attack above approximately 8 degrees, resulted in a significant performance degradation. At 256 knots IAS, a normal load of 1.2 g and MCT thrust, maximum climb performance was reduced to minus 2000 feet/min. This effect, and the associated loss of altitude, was not predicted correctly by simulation models in foregoing analyses[2, 6].

FLIGHT DATA RECONSTRUCTION AND SIMULATION

In contrast to the foregoing analysis of the flight using the DFDR, a different analysis was performed. The DFDR parameters were reconstructed by applying the DFDR pilot control inputs to an extensive model of the accident aircraft. In particular, the following issues were covered in detail by the analysis:

- Reconstruction of the flight data to review the flight from initial climb to the final flight stage
- Flight path reconstruction
- Available control margins during the flight
- Applied control inputs
- Flight control and aerodynamic contributions to the loss of control
- Initial climb performance
- Directional control system performance and effect on controllability
- Flight control capabilities
- Control loss recovery capabilities
- Manoeuvring capabilities
- Approach and landing capabilities

Flight Control and Performance

Controllability

The first notice on an engine failure will be a sudden yaw of the aircraft. If directional control is not applied, or with a fixed rudder deflection, thrust asymmetry will cause the aircraft to slip and to roll. The negative sideslip angle will create a positive rolling moment implying right wing down. Instant control compensation in an engine out condition may consist of:

- A rudder pedal input to counteract the yawing moment
- A control wheel deflection to counteract the rolling moment
- Applying a thrust reduction on the remaining engines to stop the yaw

Aircraft maneuvering in this flight condition has a direct result on the remaining control and performance capabilities of the aircraft. Turning into the direction of the remaining engines (dead engine high), creates a flight condition with more lateral margin. Bank steepening in both turn directions will cause the available performance margin to decrease.

Structural damage to the wing due a separation of the engine causes an additional lift loss and drag increase on the wing. Because these effects are a function of angle of attack, increase of angle of attack will create an additional rolling moment and yawing moment into the direction of the separated engine. This will require more opposite control wheel deflection, especially to counteract bank steepening during maneuvering.

For steady flight in the above mentioned conditions, the aircraft can be flown by:

- Reducing roll angle to zero, or
- Reducing sideslip angle to zero

Figure 8 illustrates the aircraft condition for stationary and straight flight at zero roll angle under the conditions of the accident aircraft. In wings level flight a positive sideslip angle is required for straight flight to compensate the lateral force in the vertical tailplane. This condition decreases the available performance of the aircraft due to the additional drag of the sideslip. However, more lateral control margin is created due to the contribution of the increasing negative rolling moment due to sideslip.

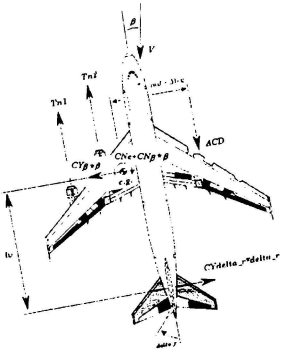


Figure 8: Straight and stationary flight with right-wing engine separation at zero roll angle

Straight and stationary flight at zero sideslip angle with a separation of the right-wing engines is illustrated in figure 9. This condition actually improves the available performance of the aircraft as the zero sideslip reduces drag. In addition, the required directional control is less demanding to sustain the flight condition. Lateral control margin is, however, reduced as no positive sideslip is available.

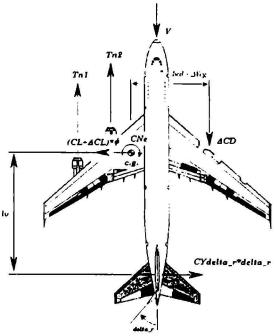


Figure 9: Straight and stationary flight with right-wing engine separation at zero sideslip angle

Performance

The simulation environment used for the analysis enables a reconstruction of the maximum performance capability of the aircraft. The maximum performance capability indicates the aircraft climb capability, for the current condition, that is available with *constant* airspeed. The actual climb rate of the aircraft may not be equal to the maximum climb capability. In this condition the aircraft acceleration is not equal to zero. The maximum performance capability is calculated by differentiation of the aircraft's specific energy[6]. Or:

$$\frac{dh_e}{dt} = \frac{dH}{dt} + \frac{V}{g} * \frac{dV}{dt}$$

Where:

$\frac{dh_e}{dt}$ = rate of change of specific energy (feet/min)

$\frac{dH}{dt}$ = altitude or climb rate (feet/min)

$\frac{dV}{dt}$ = acceleration along the flight path (feet/min^2)

g = gravitational acceleration (feet/min^2)

Simulation Environment

The simulation environment for the analysis is based on the Delft University Aircraft Simulation Model and Analysis Tool DASMAT[7]. This MATLAB®/Simulink® package was developed at the Disciplinary Group for Flight Control and Simulation

of the Faculty of Aerospace Engineering of the Delft University of Technology in order to meet the requirements for computer assisted design (CAD) and evaluation of flight control systems. The software is equipped with several simulation and analysis tools, all centered around a generic nonlinear aircraft model for state-of-the-art six-degree-of-freedom aircraft simulations. For high performance computation and visualisation capabilities, the package has been integrated as a toolbox in the computing environment MATLAB®/Simulink®. Some of the features of the package include trimming and linearization tools for linear flight control design, flight test data analysis, nonlinear on-line or off-line simulations and 3D aircraft visualisation. Applying user-generated models to the generic package customizes the software for the simulation of any specific aircraft. The Simulink® architecture of the software (figure 10) comprises three generic models of the aircraft, engine and aerodynamics. The simulation environment for the accident analysis was developed as an extension module to the DASMAT package.

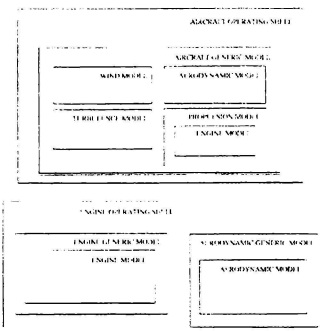


Figure 10: Model architecture of the simulation environment

The simulation environment incorporates a nonlinear aerodynamic model and a flight control system model, according to the hydraulic system architecture, of the accident aircraft. The modeled control surfaces are subjected to aerodynamic effects throughout the flight envelope. The environment is operated from its own operating shell. On-line simulations of the reconstructed model can be performed interactively or system failures can be selected that affect the aircraft's flight mechanics. For the accident analysis, the simulation environment was extended to simulate separation of the right-wing engines, incorporating all its associated system failures, and provided the capability to import DFDR pilot control inputs. In this setup, the reconstructed flight data was visualised in addition to any desired flight parameters that were beyond the scope of the original DFDR. To account for the contribution of the right-wing damage, the aerodynamic model was extended with an estimation of the aerodynamic effects due to engine separation.

Model Requirements

In general, analysis of impaired aircraft encountering one or more failure modes necessitates the definition of additional requirements to a simulation model of the aircraft. As impaired aircraft may introduce high nonlinear motions due to failures, the simulation model must comprise at least a nonlinear mathematical model in six-degrees-of-freedom. The requirements for the accident analysis resulted into the following conditions and features for reconstruction and simulation:

- Nonlinear aerodynamic model
- Flight control system model
- Simulation of engine separation and hydraulic failures
- Provisions for failure mode and effect analysis
- Simulation of all control surfaces subjected to mechanical and rate limits
- Aerodynamic effects on the control surfaces to account for actuator force limitations in the case of hydraulic power loss, including floating control devices
- Lateral control system including spoiler program
- Directional control system that simulates the upper and lower rudder independently, including actuator forces and hinge moments
- Dual yaw damper and dual ratio changer systems for failure mode conditions
- Simulation of the hydraulic system architecture
- JT9D-3 engine model modified to simulate the JT9D-7J high thrust version
- Massmodel that accounts for fuel jettison
- Process capability of DFDR data
- Visualisation of flight parameters and control surface activity
- User interface to select a desired failure mode or perform on-line simulations

Modeling data was obtained from ref. [8-11].

Operating Shell for DFDR Analysis

For the accident analysis, the software was given the capability to reconstruct the DFDR data and to perform failure mode and effect analysis. The

operating shell of the software offers several interactive capabilities for on-line simulations of the reconstructed model and analysis of the flight control system. The user may select a desired flight condition or specific failure mode scenario. Simulation results may then be recorded and used in conjunction with the analysis tools from DASMAT. This may, for instance, include the generation of a linearized model for flight control design applications.

The simulation environment for the analysis uses its own operating shell (figure 11). The environment offers the following capabilities for DFDR reconstruction and analysis of the reconstructed model:

- Import of DFDR data
- On-line simulation of aircraft, flight control and hydraulic systems
- Selection of aircraft failure modes, including:
  - Engine separation
  - Hydraulic system failures
- Visualisation of reconstructed flight data
- Visualisation of reconstructed flight control surface deflections
- 3D visualisation of reconstructed flight profile

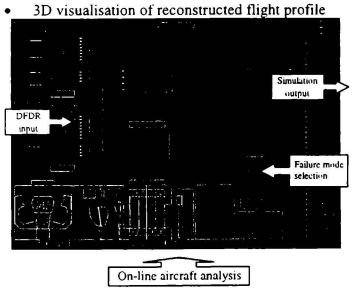


Figure 11: Operating shell for DFDR reconstruction and aircraft simulation

For the accident analysis, selection of engine separation will cause the simulation model to be configured according to the system architecture of the aircraft.

Reconstruction Setup

The reconstruction is based on a model validation method using inverse simulation[14] (figure 12). The DFDR pilot control inputs u_p are directly applied to the simulation model of the aircraft and the flight control system. The response error of the simulation output x_c and measured DFDR data x_m are input to a feedback controller. The output of the feedback controller is a measure for the fidelity of the reconstructed model. The reconstruction method has the advantage that the combined effect of structural and flight control system failures may be visualized using the simulation inputs and outputs. The estimation of the aerodynamic effects due to engine separation can be performed by adjusting the parameters of a model structure of the damaged wing until the controller output is minimized. An additional advantage of the method is that the DFDR data, with a low sample rate, can be used directly to excite the simulation model.

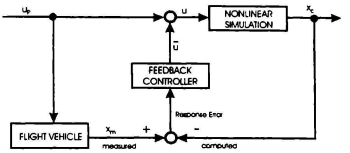


Figure 12: Inverse simulation principle for flight data reconstruction

The reconstruction setup for the analysis is illustrated in figure 13. A proportional feedback controller using pitch and roll data proved to be sufficient to obtain a reasonable match with measurements and simulation data.

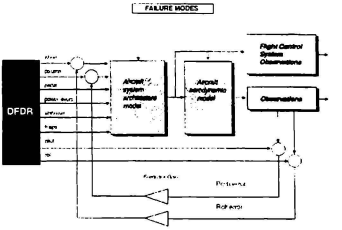


Figure 13: Setup for DFDR reconstruction and simulation

The reconstructed flight profile of the aircraft, starting from lift-off to final control loss, was divided into three separate flight legs (figure 14):

LEG 1 (t=47-371 sec): Gross takeoff flight path to engine separation

LEG 2 (t=378-647 sec): Engine separation flight path with flaps up

LEG 3 (t=648-874 sec): Engine separation flight path with flaps 1 selected

This subdivision was based on the following considerations:

- Validation of the simulation model and reconstruction method without failure modes
- Estimation of aerodynamic effects for different aircraft configurations
- Reduction of computational load

Meteorological data at the time of the crash was used during the reconstruction. The time reference during the analysis was chosen the same as in ref. [2].

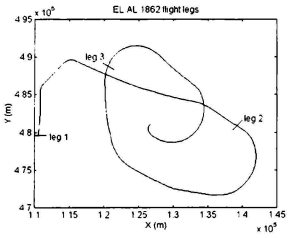


Figure 14: Flight legs for DFDR reconstruction

Aircraft Configuration

The aircraft failure mode configuration after the separation of the right-wing engines was included in the simulation and consisted of:

Aircraft systems:

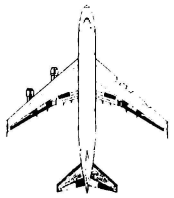
- Hydraulic systems 3 and 4 off
- Engine 1 and 2 thrust asymmetry
- Lower rudder lag

Mass properties:

- Engine no. 3 and 4 weight loss, 4,014 kg each
- Pylon no. 3 and 4 weight loss, ± 1,000 kg each
- Lateral center of gravity displacement
- Total weight loss: 10,0028 kg

Aerodynamics:

- Lift loss due to wing damage, ΔCLsep
- Rolling moment due to wing damage, ΔClsep
- Drag due to wing damage, ΔCDsep
- Yawing moment due to wing damage, ΔCnsep
- Pitching moment due to wing damage, ΔCmsep
- Right inboard aileron and spoiler 10 and 11 aerodynamic efficiency loss



Control surface lost 50% hinge moment loss/half deflection rate

Figure 15: Failure modes and damage configuration of the aircraft

RECONSTRUCTION AND SIMULATION RESULTS

Aerodynamic effects

On 31 March, 1993, a Boeing 747 Freighter encountered a separation of the no. 2 engine under turbulence conditions[13]. Despite the severe performance and controllability problems caused by the separated engine, the flight crew managed to recover the aircraft by means of an emergency landing. The flight conditions after engine separation required up to full right rudder pedal, approximately 60 degrees of right wing down control wheel deflection and overboost thrust on the no. 1 engine[6]. The structural damage to the left wing of the aircraft (figure 16) may have been representative for the amount of structural damage incurred by the aircraft of the Bijlmermeer accident.



Figure 16: Wing damage due to separation of engine no. 2, Boeing 747, 31 March 1993 (source: ref. [13])

The aerodynamic effects due to engine separation result in a lift loss and an increase of drag on the damaged wing. Consequently, an additional rolling moment due to lift loss and a yawing moment

due to the drag increase on the wing will result. At higher angle of attack these effects will further increase resulting into a reduction of controllability and performance. Figure 17 depicts the wing configuration of the accident aircraft after the separation of the right-wing engines. The figure indicates that most damage of the right wing was located near the no.3 engine. This may have caused a degradation of the aerodynamic efficiency of the inboard aileron located behind the damage. During the reconstruction, an additional pitch down moment effect was observed after the separation of the engines. It is difficult to determine the individual contributions to this effect, as the exact damage to the right wing is not known. The effect was most probably caused by a combination of a change of induced downwash near the stabilizer, wing leading edge damage, effect of the engines and effect of selected flaps. An estimate of the additional pitch down moment was required to obtain a reasonable match with the DFDR data.

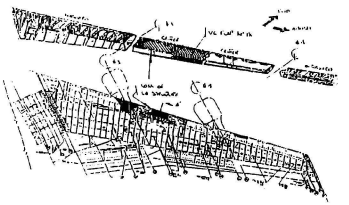


Figure 17: Wing damage configuration due to engine separation of the accident aircraft

A model structure of the aerodynamic effects due to engine separation was included for the reconstruction. The aerodynamic estimates in the model resulted into a reasonable match with the performance and control capabilities of the aircraft and DFDR data.

Figures 18-21 illustrate the effect of the aerodynamic estimates on the model input and output. Figure 18 shows the reconstructed DFDR control wheel deflection without aerodynamic effects due to engine separation. Figure 19 shows the reconstructed DFDR control wheel deflection including the aerodynamic estimates. The effect of the estimates on reconstructed DFDR roll angle is indicated in the figures 20 and 21.

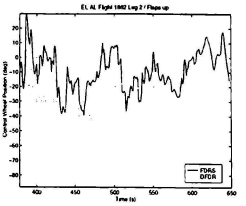


Figure 18: Reconstructed control wheel deflection without aerodynamic estimates

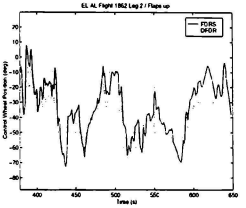


Figure 19: Reconstructed control wheel deflection including aerodynamic estimates

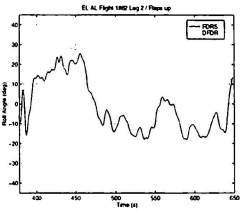


Figure 20: Reconstructed roll angle without aerodynamic estimates

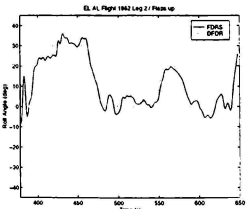


Figure 21: Reconstructed roll angle including aerodynamic estimates

Reconstructed Flight Data[3]

Leg 1 (t=47-371 sec)

Figures 22-29 illustrate the reconstruction results of the first leg at t=47-371 s. The leg covers the departure and climbout of the aircraft up to separation of the engines. The reconstructed data matches well with the DFDR data. Performance and controllability of the model are representative for the aircraft without failure modes (note: radar data was used for reconstruction of the ground track).

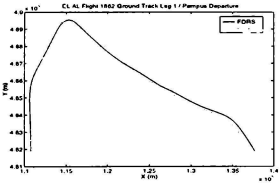


Figure 22: Reconstructed ground track

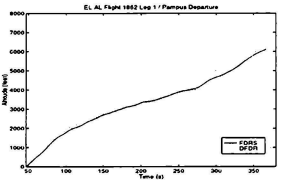


Figure 23: Reconstructed altitude

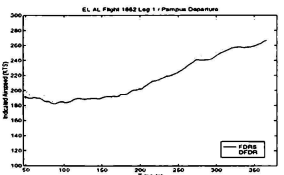


Figure 24: Reconstructed indicated airspeed

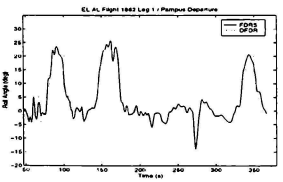


Figure 25: Reconstructed roll angle

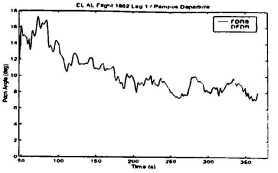


Figure 26: Reconstructed pitch angle

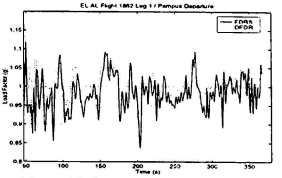


Figure 27: Reconstructed load factor

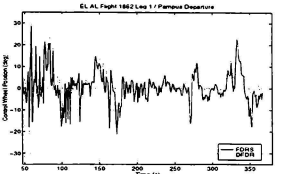


Figure 28: Reconstructed control wheel deflection

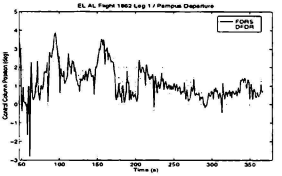


Figure 29: Reconstructed control column position

Leg 2 (t=378-647 sec)

Figures 30-37 show the reconstruction results of the second leg at t=378-647 s. The leg covers the flight after separation of the engines up to the selection of flaps 1. The reconstructed data matches well with the DFDR data. Sideslip angle is given as an example of a reconstructed parameter not available from DFDR. Performance and controllability of the model are representative for the aircraft with flaps up.

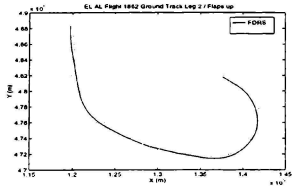


Figure 30: Reconstructed ground track

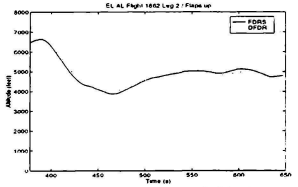


Figure 31: Reconstructed altitude

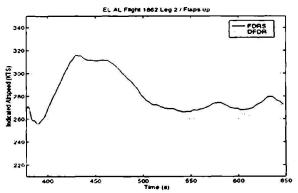


Figure 32: Reconstructed indicated airspeed

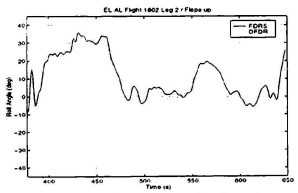


Figure 33: Reconstructed roll angle

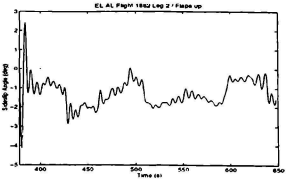


Figure 34: Reconstructed sideslip angle

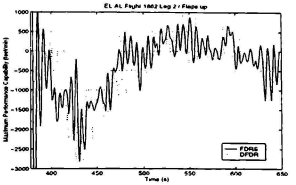


Figure 35: Reconstructed maximum performance capability

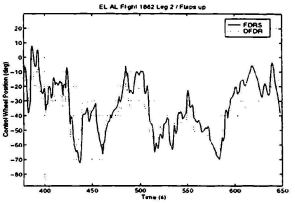


Figure 36: Reconstructed control wheel deflection

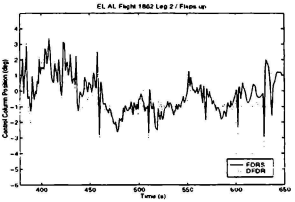


Figure 37: Reconstructed control column position

Leg 3 (t=648-874 sec)

Figures 38-45 show the reconstruction results of the third leg at time t=648-874 s. The leg covers the flight after the selection of flaps 1 up to final control loss. The reconstructed data matches well with the DFDR data. However, the impact area of the model does not match the actual impact area. Performance and controllability of the model are representative for the aircraft with flaps 1.

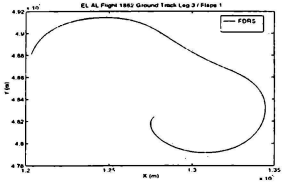


Figure 38: Reconstructed ground track

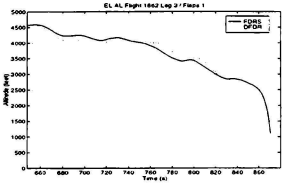


Figure 39: Reconstructed altitude

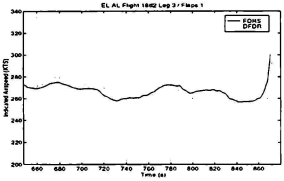


Figure 40: Reconstructed indicated airspeed

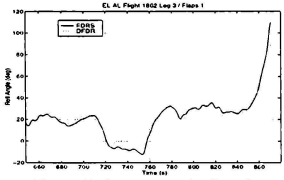


Figure 41: Reconstructed roll angle

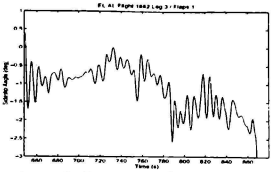


Figure 42: Reconstructed sideslip angle

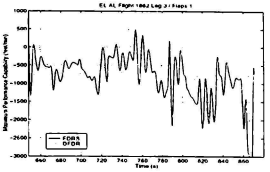


Figure 43: Reconstructed maximum performance capability

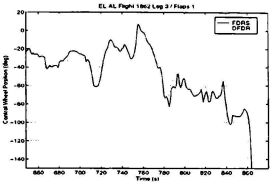


Figure 44: Reconstructed control wheel deflection

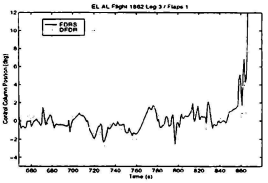


Figure 45: Reconstructed control column position

Directional Control System Analysis

The simulation model of the flight control system enabled a reconstruction and analysis of the rudder of the aircraft.

Reconstruction of the upper and lower rudder surface deflections before the separation of the engines (t=47-371 sec), indicated that reconstructed upper rudder deflection did not agree with the DFDR data (figure 48). Lower rudder activity was consistent with the DFDR lower rudder deflections (figure 49).

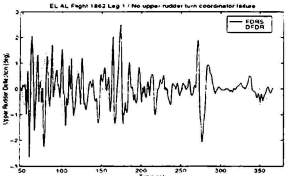


Figure 46: Reconstructed upper rudder deflection (upper rudder turn coordinator available)

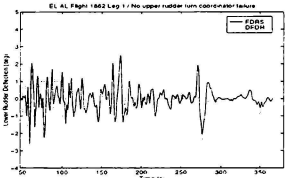


Figure 47: Reconstructed lower rudder deflection (upper rudder turn coordinator available)

At t=270 seconds, the aircraft acquires a 15 degrees roll angle to the left as a result of a control wheel input. Reconstructed upper rudder deflection is then opposite to the DFDR data. The lower rudder (figure 49) shows a reconstructed deflection consistent with the DFDR data and is opposite to the DFDR upper rudder deflection. For the upper rudder, a failure mode was applied to the simulation model. The model indicated that DFDR upper rudder activity can be reconstructed in case the upper rudder turn co-ordinator is not available (figure 50). In this condition, figure 50 indicates that at t=270 seconds the reconstructed upper rudder deflection is consistent with the DFDR data.

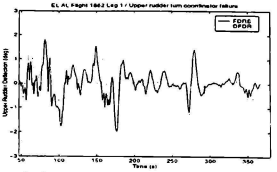


Figure 48: Reconstructed upper rudder defection (upper rudder turn coordinator not available)

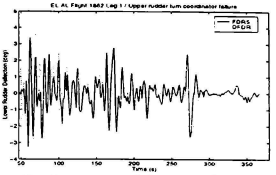


Figure 49: Reconstructed lower rudder deflection (upper rudder turn coordinator not available)

Upper and lower rudder activity was reconstructed after the separation of the engines (figures 52-56). The DFDR indicates that lower rudder authority was limited after the separation of the engines (figure 6). Rudder deflection was reconstructed for the second and third flight leg (t=378-874 s). The DFDR indicates that the deflections of the lower rudder are related to the flight condition at some stages. The most significant effect starts at t=858 s (final control loss) at full rudder pedal. At t=865 s, lower rudder deflection increases while upper rudder is limited and decreasing (figure 55).

Both upper and lower rudder are equipped with a dual-tandem actuator. Loss of one actuator, due to loss of hydraulic supply, will reduce the actuator hinge moment. Under some flight conditions, the available rudder will be limited by actuator force capability. An actuator force reduction was applied to the simulation model of the lower rudder as a failure mode. At t=480 s, full pedal is applied. In this condition, reconstructed lower rudder deflections are consistent with the DFDR data. The lower rudder is limited to about 3.2 degrees, while upper rudder is limited to 7 degrees. The reconstructed data indicated that the lower rudder deflections are primarily caused by the effect of sideslip due to thrust application. At t=590 s, engine no. 1 and 2 thrust is reduced. Sideslip decreases, which causes an increase of lower rudder authority. Flight data reconstruction indicated that the increase of lower rudder deflection at t=860 s, was

primarily caused by a combination of yaw damper commands due to yaw rate and sideslip.

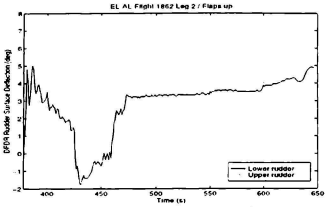


Figure 50: DFDR upper and lower rudder deflection

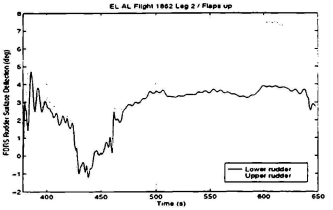


Figure 51: Reconstructed upper and lower rudder deflection

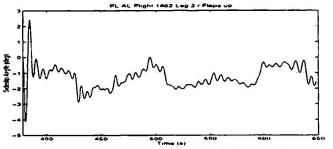


Figure 52: Reconstructed sideslip angle

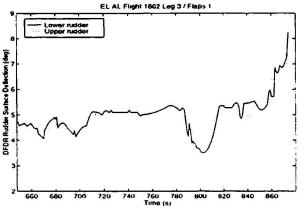


Figure 53: DFDR upper and lower rudder deflection

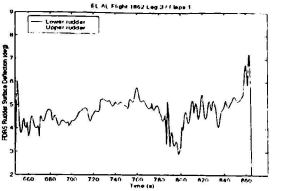


Figure 54: Reconstructed upper and lower rudder deflection: effect of yaw damper

FURTHER ANALYSIS RESULTS[3]

The additional analysis results of the reconstructed flight data can be summarized as follows:

- The reconstructed model indicated that after the separation of the engines, performance and controllability were degraded due to the weight of the aircraft and additional drag of the damaged right wing. The relative large performance degradation at angles of attack greater than 8 degrees, at a load factor of 1.2 g, and at approximately 260 KIAS, could only be reconstructed by a significant increase of drag caused by the damaged wing. Application of thrust, combined with a delayed pedal input, resulted into a loss of flight control.
- Analysis of the reconstructed model indicated that at the loss of flight control sufficient directional control margins existed to regain control of the aircraft.
- The simulation model predicted that the control loss during the last flight stage was recoverable. The control strategy for the recovery is, however, contradictory to the natural reaction of the pilot. The DFDR indicates that at final control loss, the engines are set at full thrust while the control column is pulled backwards. In addition, less than maximum rudder pedal is applied at the initiation of control loss. The simulation model indicated that for a successful recovery, engine thrust must be reduced to idle, at the expense of performance, while applying full control wheel and rudder and a forward column deflection. Applying a control loss recovery scenario to the reconstructed model for the conditions during the final stage of flight, obtained the following results:

  -Altitude loss (t=858-880 s): ≈ 800 feet
  -Control recovery roll rate (48 degrees right to 20 degrees left bank): ≈ 7.6 deg/sec
  -Maximum performance loss: ≈ -3500 feet/min
  -Application of maximum thrust (EPR 1.62 / EGT limit): ≈ 20 seconds

- The reconstructed model was able to identify that upper rudder turn coordinator was not available before the separation of the engines. It is not likely that this condition contributed to the separation of the no. 3 engine.
- The simulation model indicated that a reasonable match with DFDR lower rudder control deflections could be obtained by applying a reduction of lower rudder actuator force as a failure mode.
- Analysis of the reconstructed model indicated that straight and level flight capability existed down to approximately 250 KIAS at heavy weight and go-around thrust. A weight reduction of about 56,000 kg reduced the straight and level flight capabilities down to approximately 220 KIAS at (MCT). Weight reduction was achieved by simulation of fuel jettison up to a remaining quantity for about 20 minutes of flight. A deceleration to 240 KIAS could be made for straight flight only at heavy weight and MCT thrust.
- The reconstructed model indicated that after the separation of the engines enough controllability was available to turn in both directions.
- The simulation model predicts sufficient performance and controllability after the separation of the engines to achieve a descent at a 3.5 degrees glide slope angle for a high-speed landing or ditch at 200/210 KIAS (figure 57) and low thrust. Performance and controllability were calculated for a weight reduction of 56,000 kg due to fuel jettison. Further speed reduction below about 220 KIAS at flaps 1, resulted in a loss of go-around capabilities.

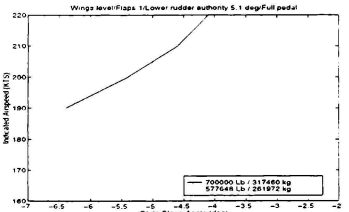


Figure 55: Simulated landing capability: attainable indicated airspeeds and flight path angles

GENERAL CONCLUSIONS.

The reconstruction method, applied for the accident analysis as described in this paper, was able to obtain a reasonable match with the DFDR data. The reconstructed model was representative for the performance and control capabilities of the aircraft. The analysis tools of the software provided the capabilities for a detailed estimate of the flying capabilities of the aircraft.

The analysis indicates that the accident aircraft was recoverable from a technical point of view. However, the required procedures to perform such a recovery are not part of current industry training practices for complex in-flight emergencies or handling qualities in degraded modes. It is therefore understandable that a successful recovery of the aircraft was highly improbable.

GENERAL RECOMMENDATIONS

Based on the results of the analysis in this paper the following general recommendations can be made:

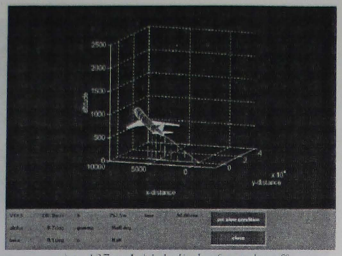
- For flight data reconstruction and simulation of aircraft accident cases, the quality of DFDR data should be improved further. To make the DFDR data more suitable for computer processing, simulation and analysis, the sample rate of the data should be increased.

Concerning a degradation of performance and controllability of the aircraft the analysis indicates that:

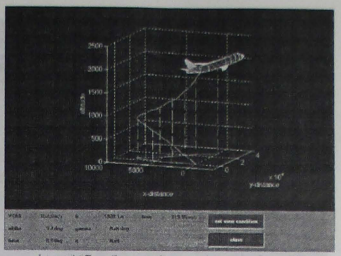
- Control of the aircraft should have a first priority.
- If control of the aircraft can still be maintained, time should be used for an assessment of the remaining performance and control capabilities.
- Unusual attitude recovery techniques may be required to regain control of the aircraft.
- Use of flaps and landing gear for configuration changes should be kept to a minimum.
- The application of thrust should be kept to a minimum.
- Further degradation of performance and controllability may be expected at the reduction of airspeed and increase of angle of attack. In these conditions, higher than normal minimum approach speeds should be considered. The selection of flaps should not be considered.

APPENDIX: 3D VISUALISATION

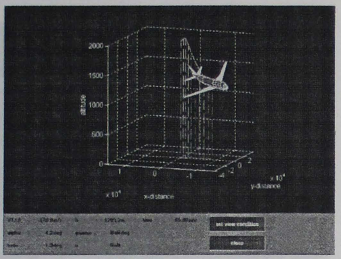
Figures a-o provide a 3D visualisation of the reconstructed flight data using the analysis software. A 3D visualisation of the reconstructed flight from take-off up to the loss of flight control is Illustrated in the figures a-k. Figures l-o show a 3D visualisation of a simulated control loss scenario. Simulation was performed using on-line analysis of the reconstructed model.



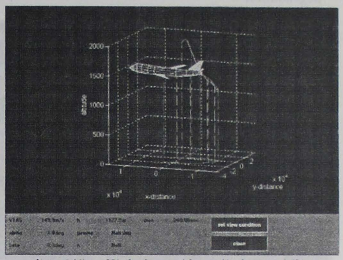
a) t=137 s: Initial climb after take-off



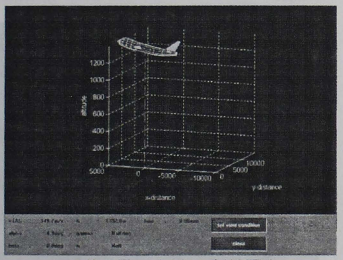
b) t=367 s: Separation engines no. 3 and 4



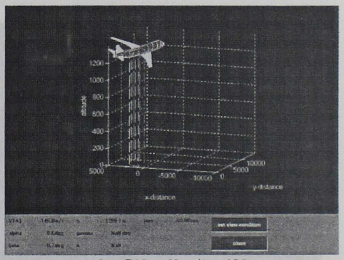
c) t=443 s: Right turn / high performance degradation



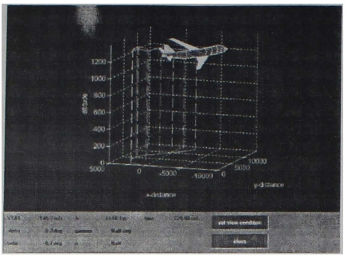
d) t=618 s: High thrust / low climb capability



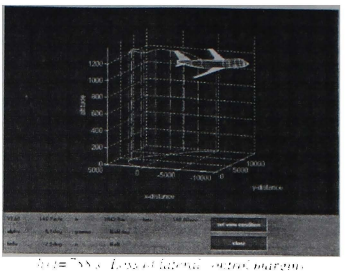
e) t=648 s: Flaps 1 selected / MCT thrust



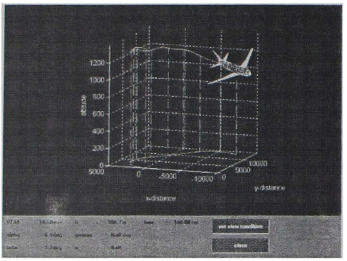
f) t=708 s: Heading 120



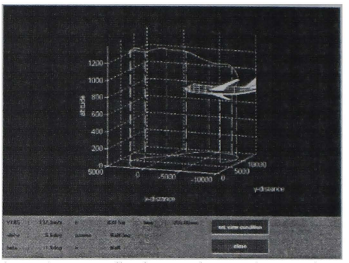
t=90 s: start of performance



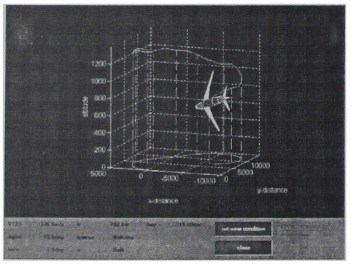
t=153 s: Loss of thrust engine number 1



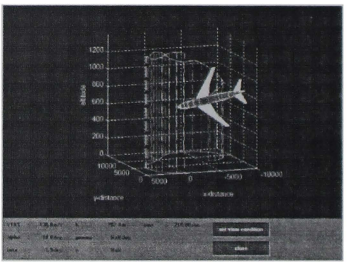
High performance degradation

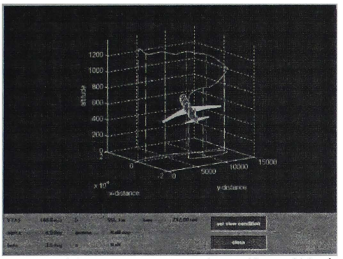


t=854 s: Application of thrust – loss of flight control

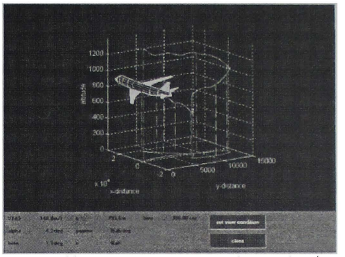


t=863 s: Engines no. 1 and 2 re-full thrust – flight control lost





a) Control loss recovery scenario: t=890 s: Altitude recovery / TOGA thrust



c) Control loss recovery scenario: t=1044 s: Steady climb / MCT thrust

REFERENCES

1. *EL AL Flight 1862*, Aircraft Accident Report 92-11, Netherlands Aviation Safety Board, Hoofddorp, 1994.
2. *EL AL Flight 1862. Results of Flight Data Recorder Analysis and Piloted Simulations, Boeing*, May 17, 1993.
3. Smaili, M.H., *Flight Data Reconstruction and Simulation of EL AL Flight 186*, graduation report, Delft University of Technology, Delft, 1997.
4. *Flight Data Recorder Factual Report*, Office of Research and Engineering, National Transportation Safety Board, Washington, D.C., 1992.
5. *Een Beladen Vlucht*, Report Dutch Parliamentary Inquiry, SDU 1999.
6. Broos, P.M., *EL AL Flight 1862 Performance Capability Evaluation*, June 1993.
7. Van der Linden, C.A.A.M., *DASMAT- Delft University Aircraft Simulation Model and Analysis tool*, LR-781, Delft University of Technology, Delft, 1996.
8. *The Simulation of a Large Jet Transport Aircraft, Vol. I: Mathematical Model*, NASA CR-1756, March 1971.
9. *The Simulation of a Large Jet Transport Aircraft, Vol. II: Modeling Data*, NASA CR-114494, September 1970.
10. *Boeing 747 Aircraft Operations Manual*, Royal Dutch Airline, 1976.
11. *Boeing 747 Maintenance Manual*, Royal Dutch Airlines.
12. Fischenberg, D., *Ground Effect Modeling Using a Hybrid Approach of Inverse Simulation and System Identification*, AIAA Modeling and Simulation Technologies Conference, Conference Proceedings, Portland, Oregon, 1999.
13. *In-Flight Engine Separation Japan Airlines, Inc, Flight 46E, Boeing 747-121, N473EV Anchorage, Alaska, USA*, NTSB/AAR-93/06

A00-37298

AIAA Modeling and Simulation Technologies Conference and Exhibit
14-17 August 2000 Denver, CO

AIAA-2000-4587

FBSS-RAIS
Flight Crew Behavior Simulation System
-Reconstruction of Accident / Incident Scenario

Koji Muraoka and Noriaki Okada*
National Aerospace Laboratory (NAL), Tokyo, Japan

ABSTRACT

The National Aerospace Laboratory (NAL) has developed a software tool which can be utilized for human factors analysis of an accident or incident. This software enables timeline analysis of flight crew behavior based on a human performance database, and incorporates both an ergonomic flight crew model and a cockpit model. A scenario of flight crew actions derived from CVR (Cockpit Voice Recorder) and DFDR (Digital Flight Data Recorder) data is used to drive a simulation, the results of which are provided in the form of a three-dimensional animation with additional verbal communication and flight status information.

INTRODUCTION

It is reported that more than 70% of recent aircraft accidents involve flight crew human factors issues[1]. The crashes of an Airbus A300-600R at Nagoya, Japan in 1994 and of a Boeing B757 near Cali, Columbia in 1995 are typical accidents of which the causes were due mainly to improper flight crew actions in operating the airplane's automatic systems. In such accidents, examination of flight crew behavior is an important issue.

A software package "FBSS-RAIS" (Flight crew Behavior Simulation System–Reconstruction of Accident/Incident Scenario) has been developed by the National Aerospace Laboratory (NAL) for the analysis of flight crew behavior in an accident or incident. FBSS-RAIS generates a sequence of flight crew behaviors as a three-dimensional animation with additional verbal communication and flight status information. The task analysis data and animations provided by FBSS-RAIS support the examination of flight crew behavior. While a software tool is often used to visualize DFDR (Digital Flight Data Recorder) flight path data[2], FBSS-RAIS can be used to construct animations of flight crew behavior from DFDR and CVR (Cockpit Voice Recorder) records.

FBSS-RAIS was developed for release to end users and is based on a prototype flight crew behavioral model[3,4,5]. Research on flight crew behavior simulation conducted at NAL aims to develop a technological foundation for dealing with flight crew human factors issues using computers. The flight crew model enables simulations that include both human physical and cognitive characteristics and can be used to carry out pre-evaluation and pre-analysis associated with flight crew human factors prior to piloted flight simulations and flight tests. The model was constructed to facilitate analysis from an operational point of view rather than for the development of cockpit hardware, and is aimed at evaluating newly-developed procedures, piloted flight simulation scenarios, reconstruction of crew behavior during flights in which accidents or incidents have occured, and so on.

So far, the prototype model has demonstrated its capabilities in simulating the performance of nominal flight procedures and abnormal events as well as flight crew actions. The results of simulations are provided in the form of three-dimensional animations that can be used in time-line analysis. For accident analysis, the applicability of the model was demonstrated by applying the framework of its flight crew action simulation functions to the reconstruction of an accident scenario which was derived from CVR and DFDR analyses contained in the accident report on the crash of a China Airlines Airbus A300-600R at Nagoya, Japan in 1994. FBSS-RAIS was developed by adding a graphical user interface and functions for handling various database files to the prototype model.

This paper describes the detailed structure of FBSS-RAIS and example applications to accident analysis. The capabilities of the model, possible expansion of its functions, and further applications are also discussed.

The FBSS-RAIS Package

ARCHITECTURE

Figure 1 depicts the architecture of the FBSS-RAIS software. FBSS-RAIS constructs flight crew behaviors based on a flight crew model and a cockpit model. Both of these models refer to databases of human body size, human performance and cockpit layout. The tasks of the flight crew model are specified in a flight crew action scenario file, and flight profile is specified by a flight profile scenario file. Users prepare these scenario files from CVR tran-

* Researcher, National Aerospace Laboratory

Copyright © 2000 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

scripts, DFDR data and other information when conducting behavioral simulations. Users can also tailor the databases to fit the case under investigation.

To perform a behavioral simulation of a two-crew cockpit, users set up the conditions by assigning a set of seven files: scenario files of flight crew behavior and flight profile, and database files of human body size and human performance for each flight crew member's model and of cockpit layout (Fig. 2). Simulation results are output in the form of three-dimensional animations as well as data plots of flight parameters and simulated flight crew actions. Later sections describe the functions of each element in detail.

PLATFORM

FBSS-RAIS runs on the operating systems IRIX ver. 5.3 or later and Solaris ver. 2.4 or later. A speech function is also available if the True Talk Speech Server™ is installed.

SCENARIO FILES

(1) Scenario of Flight Crew Actions

This scenario file (Figure 3) is prepared as a text file. Flight crew tasks are specified by time (when an action was performed), subject (who carried out the action), verb (type of action/action command, listed in Table 1), and object (for what or to what was the action performed) as natural English sentences. Initial conditions of each flight crew member such as role (PF or PNF), seat (left or right), and flight phase (approach, cruise, etc.) are also specified in the scenario file. For example, the sentence

00:01:10.00, FO, set, flap_lever, 30,

specifies that the first officer model commence setting the flap lever to 30 degrees at 0:01:10.

Commands can also be specified in a past tense form using an "ed" notation (Table 1). These commands specify the time of completion of the action. For example, the sentence

00:01:10.00, FO, seted, flap_lever, 30,

specifies that the first officer model start setting the flap lever to 30 degrees such that the task is accomplished at 0:01:10.

In an accident analysis, cues for inferring crew behavior are provided by data from the DFDR and the CVR. In particular, DFDR data can indicate the crew actions which result in changes of system status. Action commands which can predict the timing of these actions are useful in such analyses.

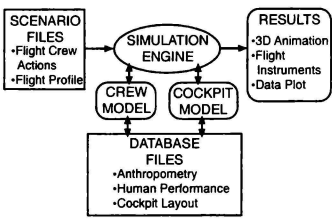


Figure 1. Software Architecture

Table 1. List of Subjects and Commands of Actions

(a) List of Subjects

Subjects	
CAP	Captain
FO	First Officer
ATC	Air Traffic Controller
SYS	System Messages

(b) List of Commands of Actions

Verbs	Past Tense	Actions
touch	touched	Touching an object.
push	pushed	Pushing switches.
set	seted	Set an object to a certain status.
take_over	-	Taking over control.
say	-	Saying words.
call	called	Communicate with ATC.
glance	glanced	Taking a glance.
look	looked	Looking at the object
check	checked	Checking or reading an indicator.

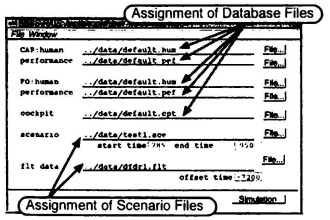


Figure 2. FBSS-RAIS Menu window
(Assignment of a simulation case)

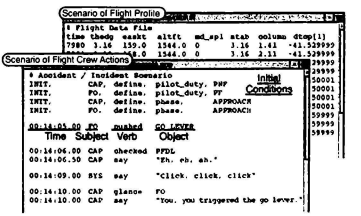


Figure 3. Scenario Files

The detailed logic of action commands is described in a later section.

(2) Scenario of Flight Profile

The flight profile scenario is specified as a text format table of parameters with time tags. The parameters are connected with the indications of the flight instruments and with the movement of cockpit devices such as control columns. The names of parameters are defined at the top of their columns of data.

DATABASE FILES

Database files (Figure 4) are referred to by the cockpit and human agent models. FBSS-RAIS reconstructs flight crew behavior under constraints indicated in these database files. This section describes the contents of each database file. The detailed structure of the models is described in a later section.

(1) Anthoropometric Database File

The stature of the flight crew model and ratios of parts of the body against stature are defined in the anthropometric database file. The human flight crew model calculates the attitude of the body based on the sizes of body parts derived from this database. It is usually easier to obtain information on flight crew stature and weight rather than on the sizes of parts of the body, such as the lengths of the lower arms. By using standard ratios of body part sizes against stature, it is easy to obtain a reasonable approximation to the body of a flight crew member by varying only stature.

(2) Human Performance Database File

This specifies speeds of moving the hands, the rotational speed of the neck, limit angles of joints, and the times required to complete tasks such as setting a two-way toggle switch. Task times for each action are calculated based on these data. Users can vary these data to obtain simulations of fastest human performance or of average performance.

(3) Cockpit Layout File

This specifies the sizes and locations of components of the cockpit. A variety of cockpit layouts for different types of aircraft can be defined by editing this database.

COCKPIT MODEL

The FBSS-RAIS cockpit model comprises simplified shapes of cockpit elements such as flap lever, control column wheel, switches, panels and so on. On each of these elements is defined a point that may be touched by the flight crew models' hands. Elements such as the throttle levers, flap lever, gear lever, control column, and switches can be associated with a flight parameter (e.g. the angle of the column control, or gear up or down status). The layout of a particular cockpit is specified by the cockpit layout file.

FLIGHT CREW MODEL

FBSS-RAIS is not intended to be used for detailed ergonomic analysis such as workspace design, but is aimed at the visualization of sequences of flight crew behavior and timeline analysis based on a human performance database. Therefore, simplified models of the human body structure and auditory speech functions to simulate communication in the cockpit are implemented. Also, action commands are implemented to allow the simulation of cockpit tasks.

(1) Human Body

Figure 5 illustrates the structure of the human crew model which is represented in the form of a skeletal linkage system. Skeletal components are assumed as rigid rods with no thickness or mass. These are connected by six joints: pelvis, neck, and both shoulders and elbows. Each joint allows only rotational motion. Joints in the lower half of the body are fixed, since most flight crew actions in the cockpit are performed in the sitting position. The sitting position and sizes of each part of the body are defined in the anthropometric database file. Reference points are added to

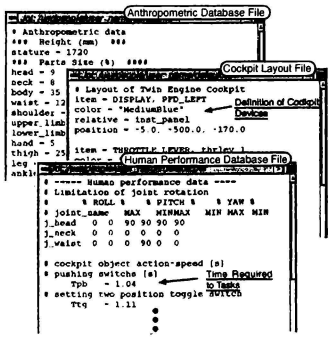


Figure 4. Database Files

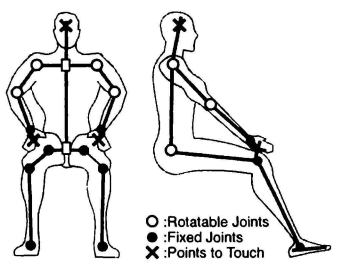


Figure 5. Structure of the Flight Crew Model

each hand and to the eye positions to enable the model to manipulate and look at cockpit components.

An algorithm for calculating the angles of each joint for touching an object is as follows. First, the hand to be used is selected by judging which shoulder is closer to the object. The pitch and yaw angles of the shoulders are then calculated to orient the hand towards the object. If it is not possible to reach the object only by stretching the arm with the torso upright, the pitch angle of the pelvis is also calculated. The shoulder and elbow roll angles are then calculated to adjust the distance from the hand to the object. The angles of the neck are calculated to orient the face perpendicular to the direction from the head to the object.

Although the human physical model is composed of a skeletal linkage system, the model is presented in the graphic display as a thee-dimensional solid body constructed of polygons.

(2) Speech Function

An auditory speech function is available via a separate speech server program which can synthesize speech from text sentences sent from client programs. Users can specify speech rate and vocal characteristic parameters such as vocal tract size, open quotient, aspiration, and spectral tilt of the glottal source, in the human performance database file. Voice parameters are set arbitrarily for each model so that the speakers' voices may be distinguished as those of the captain, first officer and air traffic controller.

(3) Flight Crew Actions

Flight crew actions are calculated based on the following assumptions.

(a) A simulated flight crew member moves its hands in a straight line at a constant speed specified in the human performance database file.

(b) A simulated flight crew member returns its hands and neck to their initial positions upon completing an action.

(c) A simulated flight crew member rotates its neck with a constant speed specified in the human performance database file.

(d) A simulated flight crew member looks at an object with its face turned perpendicular to it.

Commands associated with the past tense predict the starting time of the action by calculating the time required to accomplish the action based on the current positions of the hands.

If the simulated crew member cannot complete a specific action in time due to joint rotation or hand movement speed constraints, FBSS-RAIS reports an error message. For example, if the simulated flight crew member seated in the left seat tries to push a switch located on the right side of the central pedestal, it will be unable touch it because of joint rotation limitations. Also, a simulated flight crew member will be unable to complete both actions of setting flap and gear levers within one second unless unusually short task times are defined in the human performance database.

The completion of an action is judged from whether or not the task of the action is accomplished (e.g. whether a switch has been pushed or flap lever has been set) while the simulated flight crew member moves its hands from the control back to the initial position. An action might be interrupted before its completion by another action, and this would give rise to an inconsistency in the scenario because the simulated flight crew member would be unable to

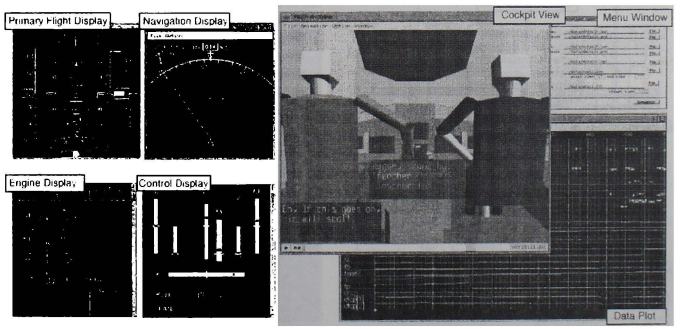


Figure 6. FBSS-RAIS Display Image

perform the prescribed actions in time.

PRESENTING SIMULATION RESULTS

Simulation results are presented as three-dimensional animations of the cockpit, reconstructions of the flight displays, and data plots of flight parameters and flight crew actions. Figure 6 shows a display image of FBSS-RAIS.

(1) Three-Dimensional Animation

A sequence of flight crew behaviors is shown as a real-time animation. Users can adjust the virtual camera orientation and position through a graphical user interface. It is possible to check a particular action by zooming the camera to specific devices, such as thrust levers or an autopilot switch. Cockpit communications are shown as a superimposed text dialog in addition to the speech function.

(2) Flight Instruments

Parameters from the flight data scenario file are shown on reconstructed PFD (Primary Flight Display), ND (Navigation Display), engine display and control displays. These are synchronized with the animation of flight crew behavior.

(3) Data Plot

Time histories of the simulated flight crew actions as well as flight parameters are shown. These data plots are useful for examining the timing of actions in detail.

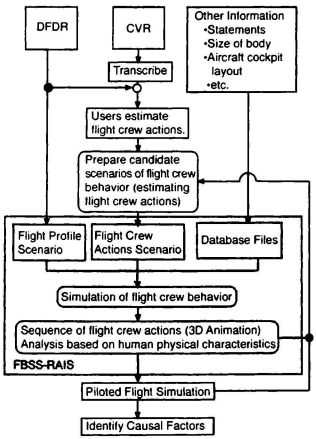


Figure 7. Typical Process of Reconstructing Flight Crew Actions

RECONSTRUCTION OF FLIGHT CREW BEHAVIOR

PROCESS OF RECONSTRUCTION

Figure 7 illustrates the typical process of reconstructing the flight crew actions in an accident or incident by FBSS-RAIS. In most cases, the DFDR and CVR data can be utilized to analyze an accident along with other information which are often also available such as post-crash interview statements and anthropometric data such as crew member heights. Using such information, an investigator creates possible candidate flight crew action scenarios and prepares database files of anthropometry and cockpit layout. Recorded flight data can be directly translated into flight profile scenarios. Based on these scenarios, FBSS-RAIS can generate animations of flight crew behavior and enable timeline analysis of flight crew actions to be performed before piloted flight simulations are conducted.

EXAMPLE APPLICATIONS

To demonstrate the utilization of the tool, the behavior of the flight crew in the 1994 crash of a China Airlines Airbus Industrie A300-600R at Nagoya airport is reconstructed in this section. Analyses of specific actions in the accident were also conducted.

Reconstruction of Flight Crew Behavior

(1) Summary of the Accident

According to the accident report[6], the events leading up to the crash began while the aircraft was approaching Nagoya airport for landing. While the flight officer was flying a manual approach (not with autopilot), he inadvertently operated the GO lever located on the forward side of the throttle levers, which caused the AFCS (Automatic Flight Control System) to enter "go-around" mode. The flight crew struggled with the AFCS as they continued the approach and as a result, the horizontal stabilizer moved to its full nose up position, contrary to the intention of the flight crew. This caused an out-of-trim situation as a consequence of which the airplane stalled and then crashed. It was reported that the flight crew was confused by the mode logic of the

Table 2. Part of Scenario of Actions

```
# Accident Scenario Nagoya CAL
INIT,   CAP,   define,  pilot_seat,      left
INIT,   CAP,   define,  pilot_duty,      "PNF"
INIT,   CAP,   define,  flight_phase,    APPROACH
INIT,   FO,    define,  pilot_seat,      right
INIT,   FO,    define,  pilot_duty,      "PF"
INIT,   FO,    define,  flight_phase,    APPROACH

00:13:57.00,   CAP,   say,      "All lights on."
00:14:05.00,   FO,    pushed,   golever1   # FO hit the GO Lever
00:14:06.00,   CAP,   checked,  PFD_L
00:14:06.50,   CAP,   say,      "Eh, eh, eh."
00:14:09.00,   SYS,   say,      "Click, click, click"
00:14:10.00,   CAP,   glance,   FO
00:14:10.00,   CAP,   say,      "You, you triggered the go lever."
```

autopilot.

(2) Scenario Files

Scenario files of flight crew actions and of flight parameters were prepared from the results of the analyses of the CVR and DFDR contained in the accident report. The scenario files were prepared under the following assumptions:

(a) An action for which a cue was obtained from the CVR was assumed to start at the time when the conversation started.

(b) An action for which a cue was obtained from the DFDR was assumed to be completed at the time when the system status changed. Commands in the past tense form were used for these actions.

(c) All of the available flight parameters were tabulated simultaneously at one second intervals.

A part of the scenario is shown in Table 2.

(3) Database Files

The analysis was not aimed at conducting an accident investigation, but to demonstrate the applicability of FBSS-RAIS to accident analysis. Thus, it was decided to use the dimensions of a general-purpose research simulator and standard anthropometric and human performance data[7,8] for this simulation.

Two files of human body size were prepared, one for each of the flight crew members. The height of each flight crew member was set in each file, but the same human performance database file was used for both flight crew members. The cockpit layout file was prepared to simulate the layout of the A300-600R cockpit.

(4) Simulation Results

The results showed no simulation-generated errors or inconsistencies, i.e. all of the tasks prescribed in the scenario could be accomplished by either the captain or the first officer model within their constraints. As shown in Figure 8, the sequence of the flight crew actions was output in the form of a three-dimensional graphical animation with synthesized voices.

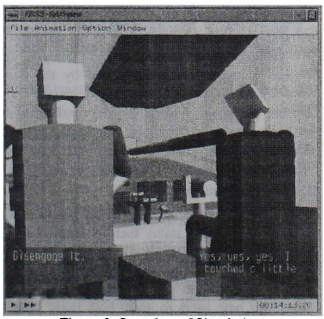


Figure 8. Snapshot of Simulation
(First officer pushed LAND switch at 14'13".)

Analysis of Flight Crew Behavior

(1) Scope of the Analysis

The fact that autopilots 1 and 2 were engaged by someone while the crew was struggling with the AFCS was recorded in the DFDR data. However, as no call-outs nor instructions regarding this action were found in the CVR record, it was difficult to determine who conducted this action and why. In this section, each possibility, i.e. that either the captain or the first officer engaged the autopilots, was examined by preparing two sets of scenarios. If the action of engaging the autopilots were to be interrupted by another action before or after it, this would determine who conducted the action based on the physical characteristics of the flight crew model.

(2) Scenario and Database

Two actions scenarios were prepared for the analysis. In scenario No.1, the captain's model engaged the GO lever, while the first officer's model engaged it in scenario No.2. The same flight profile scenario and database files were utilized for each simulation.

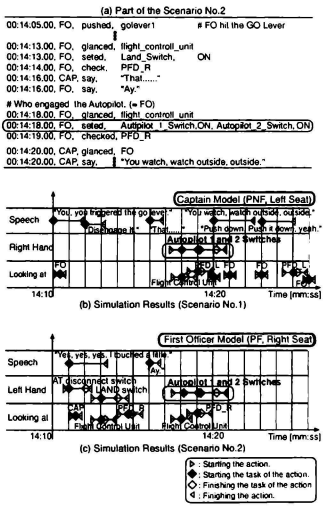


Figure 9. Simulation Results

(3) Simulation Results

Figure 9 shows the time history of each simulation. The results indicated that each of the scenarios has no inconsistency with actions before and after the autopilot was engaged. This indicates that both scenarios are possible under the conditions of the simulation such as human body size and human performance. The physical characteristics of the FBSS-RAIS human model could not determine which of the scenarios was correct, as described in the accident report and in reference 4.

DISCUSSION

APPLICATION TO ACCIDENT ANALYSIS

The example applications presented in the preceding section demonstrate the process of reconstructing flight crew behavior and the use of the results in the analysis of an accident. The advantages of using FBSS-RAIS for accident analysis can be summarized as:

(1) Intuitive sequence of flight crew behavior: FBSS-RAIS provides an intuitive image of flight crew behavior by three-dimensional animation. Animations could be used to share a common image of the crew behavior sequence among members of an investigation team during each phase of accident analysis.

(2) Analysis based on human physical characteristics: During the course of an investigation, especially in the initial phase, many factors are considered and a variety of candidate scenarios must be examined. Although in the above example FBSS-RAIS could not determine which crew member engaged the autopilot, FBSS-RAIS has the potential to exclude unfeasible scenarios on the basis of human performance, anthropometry, and cockpit layout, providing data as to whether the simulated flight crew can accomplish each prescribed task within specified times. Also, a single scenario can be examined in detail by varying parameters such task speeds in the human performance database.

When examining flight crew behavior in an accident, in addition to the crew behavior sequence itself it is necessary to consider all the pertinent elements such as weather conditions, system failures, flight procedures, and aircraft flight characteristics. The current version of FBSS-RAIS includes neither a flight dynamics model, nor a systems function model, nor a human cognitive model. If such models, which have been developed in the prototype model, were incorporated, FBSS-RAIS would be able to provide sequences of behaviors for non-normal aircraft operating procedures as well as for normal ones. This would enable the comparison of flight crew behaviors in nominal situations with those of accident cases. Further possible functions to enhance the capabilities of FBSS-RAIS are discussed below.

APPLICATION AREAS

In addition to accident analysis, there are further potential applications of FBSS-RAIS such as:

(a) Effective Feedback of Incident and Accident Information

Feedback of information on accidents or incidents to pilots is an effective method of enhancing flight safety, and such many programs, for example FOQA (Flight Operations Quality Assurance), are conducted by major airline companies. Visualization software which provides flight profile animations are sometimes utilized for such purposes. FBSS-RAIS could be used to emphasize events in which flight crew human factors are concerned.

and,

(b) Utilization as a Training Material

Information regarding an incident or accident could be used as case study material for human factors training. Using FBSS-RAIS, a three-dimensional animation of flight crew behavior can be reconstructed more easily than a movie using human actors in a simulator cockpit. Also, a more intuitive image of behavior sequences can be provided than is possible with handouts of CVR transcripts and flight profile plots.

FURTHER FUNCTIONS

This section introduces possible enhancements to FBSS-RAIS which would enable more detailed analysis of accidents and incidents.

(a) More Detailed Animations

FBSS-RAIS provides a three-dimensional animation with simplified human posture and cockpit layouts. Adding a more detailed biomechanical model and a more realistic presentation of the human body will provide a more realistic and impressive image of flight crew behavior. However, the authors consider the animations produced by the current version of the software to be sufficient for the purposes of accident or incident analysis, since objective images can be obtained by the current model and tasks are reconstructed mainly by the human performance database. More detailed animations would be useful for emphasis when used for the feedback of accident or incident information or in human factors training materials.

(b) Flight Path Animation

Flight crew behavior should not be examined apart from aircraft flight status such as attitude and acceleration. While FBSS-RAIS shows the aircraft state by reconstructing flight instruments, the addition of functions to generate flight profile animations will help users to grasp both flight crew behavior and flight status more intuitively. NAL has been developing a flight data visualization program "DRAP" (Data Review and Analysis Program),

which is aimed to be used for feedback of flight data to pilots in an FOQA program, and DRAP's flight path animation functions will be incorporated into FBSS-RAIS.

(c) More Detailed Anthropometric Model

Since workspace analysis and collision detection are not always necessary for accident analyses, FBSS-RAIS incorporates a simplified skeletal linkage system for the anthropometric model. As discussed above, a more detailed anthropometric model that incorporates more joints, skeletal elements, mass and forces would provide not only a more human-like image but also more detailed reconstructions based on a biomechanics. However, task analysis based on empirical data is considered more practical than analysis based on analytical human models, especially for the reconstruction of flight crew behavior. In this regard, accumulating performance data such as task times in non-normal and normal situations would be more meaningful for analysis.

(d) Cognitive Functions

In the example analysis in section 3, FBSS-RAIS could not determine which crew member engaged the autopilots. Since the problem involved the interaction of the crew with the autopilot, a cognitive model that can simulate a pilot's understanding of the autopilot system and actions taken after the occurrence of unusual events could help produce a more detailed analysis.

We have developed a prototype model which has cognitive functions for simulating normal and non-normal procedures. Since this model was developed to analyze flight crew behavior in 'nominal' situations, it is not possible to reconstruct pilot mental processes in such specific accident cases. Preparing more sophisticated mental models that contain hierarchies of information processing and activities and performing iterative calculations such as fault tree analyses may also generate sequences of flight crew behavior.

(d) Flight Dynamics and Aircraft System Model

The prototype model of references 4 and 5 incorporates flight dynamics and aircraft system simulation functions. Utilizing these functions, examinations of nominal procedures, flying characteristics and system functions could be performed, and it would be possible to compare a reconstructed sequence of flight crew behavior with nominal procedures. Also, the nominal logic of systems such as autopilot modes could be confirmed using a desktop computer. However, before such models can be utilized for accident investigation, it must be verified that their behaviors match precisely those of the actual systems.

CONCLUDING REMARKS

A software tool "FBSS-RAIS" has been developed that enables the reconstruction of flight crew behavior in an accident or incident. The characteristics of this tool are summarized as follows:

(1) An intuitive sequence of flight crew behavior is provided in the form of three-dimensional animation.

(2) Flight crew behavior is reconstructed from a scenario which specifies the actions of the flight crew. Scenarios are prepared by users based on DFDR data and CVR transcripts.

(3) FBSS-RAIS enables timeline analysis based on an anthropometric database, a human performance database and cockpit layout.

Since this software is developed to be released to end users, we are willing to supply the program to any organization. We are also prepared to conduct analyses of flight crew behavior in an accident utilizing FBSS-RAIS, should the Japanese Aircraft Accident Investigation Commission request investigation support. Fortunately, we have not had an accident in Japan for which FBSS-RAIS can be used, so far.

REFERENCES

[1] FAA Human Factors Team; The Interface Between Flight crews and Modern Flight Deck Systems, 1996.

[2] Soekka, H. (Ed.); Aviation Safety -Human Factors - System Engineering - Flight Operations - Economics - Strategies - Management -, VSP, 1997.

[3] Muraoka, K.; Computational Modeling of Crew Behaviors, Proc. of 34th Aircraft Symposium, pp.104-107, Oct. 1996.

[4] Muraoka, K.; Crew Behavioral Model for Timeline Analysis, 10th CEAS Conference "Free Flight", 1997.

[5] Muraoka, K. and Okada, N.; Crew Behavioral Model for Evaluating Procedures of Research Aircraft, SAE-985581, 1998.

[6] Aircraft Accident Investigation Commission Ministry of Transport; Aircraft Accident Investigation Report, China Airlines, Airbus Industrie A300-B4-622R, B1816, Nagoya Airport, April 26, 1994, July 19, 1996.

[7] Aeromedical Laboratory, Air Development & Test Wing JASDF; Anthropometry of JASDF Personnel and its application for human engineering, 1988.

[8] James L. Sundstrom; NASA TLA Workload Analysis Support , Volume 1 - Detailed Task Scenarios for General Aviation and Metering and Spacing Studies, NASA CR 3199,1980.

[9] Anon.; Designer's Guide to Human Performance Modeling, AGARD-AR-356, 1998.



For permission to copy or republish, contact:
American Institute of Aeronautics and Astronautics
1801 Alexander Bell Drive
Suite 500
Reston, VA 20191–4344

ISBN
1-56347-378-X